Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Návrh a implementace informačního systému hokejbalových hráčů

Alan Miškovský

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Alan Miškovský

Systémové inženýrství a informatika
Informatika

Název práce

Návrh a implementace informačního systému hokejbalových hráčů

Název anglicky

Design and Implementation of Information System of Hockeyball Players

---

Cíle práce

Cílem bakalářské práce je vytvořit funkční informační systém hokejbalových hráčů. Systém bude vytvořen jako webová aplikace. Dále k otestovaní správné funkčnosti systému budou využity automatické testy, které budou vytvořeny v programu Pycharm s využitim robot framework.

Metodika

1. Na základě studia odborných zdrojů popište formou literární rešerše doménu vývoje webových aplikací dle osvědčených metod soGwarového návrhu

2. Proveďte analýzu uživatelských požadavků na webovou aplikaci a popište je vhodnými nástroji (Use Case)

3. V souladu s doporučenými postupy soGwarového inženýrství vytvořte návrh vlastní webové aplikace (konceptuální model)

4. Návrh aplikace implementujte ve vhodném prostředí a řádně otestujte pomocí automatických testů vytvořených v programu Pycha

Doporučený rozsah práce

30-40 stran

Klíčová slova

pycharm,phpstorm, robot framework, informační systém, webová aplikace

---

Doporučené zdroje informací

KOSEK, J. *PHP – tvorba interaktivních internetových aplikací : podrobný průvodce.* Praha: Grada, 1999. ISBN 80-7169-373-1.

Luke Welling, Laura Thomson. Mistrostvní PHP a MySQL. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.

Předběžný termín obhajoby

2020/21 ZS – PEF (únor 2021)

Vedoucí práce

Ing. David Buchtela, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 14. 3. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 14. 3. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 10. 03. 2021

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Návrh a implementace informačního systému hokejbalových hráčů" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2021 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Davidu Buchtelovi, Ph.D. za pomoc, dohled a cenné rady při psaní této práce.

Návrh a implementace informačního systému hokejbalových hráčů

Abstrakt

Tato bakalářská práce je zaměřena na návrh, vytvoření a otestování informačního systému pro evidenci hokejbalových hráčů. Teoretická část je zaměřena na seznámení s možnými metodiky pro vývoj softwaru a jednotlivými fázemi vývoje. Následuje popis MVC vzoru a představení API. Teoretická část je zakončená krátkým popisem SQL Injection. Praktická část se věnuje celému průběhu vývoje informačního systému. Na začátku jsou představeny technologie, které byly využity při vývoji aplikace. Velká část se věnuje popsání funkcionalit za pomocí use case spefikací, za kterými následuje detailní popis wireframe modelů. Po popisu wireframe modelů následuje představení celé implementace aplikace, zakonečné představením databázové vrstvy. Poslední část se věnuje testování jednotlivých funkcionalit, přičemž základní funkcionality byly otestovány za pomocí automatickýh testů.

Klíčová slova: robot framework, informační systém, webová aplikace, softwarové testování

Design and Implementation of Information System of Hockeyball Players

Abstract

This bachelor thesis is focused on the design, creation and testing of an information system for the evidence of hockeyball players. The theoretic part is focused on getting acquainted with possible methodologies for software development and on individual stages of development. The following is a description of the MVC pattern and an introduction to the API. The theoretical part ends with a short description of SQL Injection. The practical part deals with the whole course of information system development. At the beginning, the technologies that were used in the development of the application are introduced. A large part is devoted to the description of functionalities using use case specifications, followed by a detailed description of wireframe models. The description of the wireframe models is followed by a presentation of the entire implementation of the application, ending with a presentation of the database layer. The last part is devoted to testing individual functionalities, while the basic functionalities were tested using automatic tests.

Keywords: robot framework, information system, web application, software testing

Obsah

Seznam obrázků

Seznam ukázkových kódů

Seznam použitých zkratek

XML – Textový značkovací jazyk sloužící pro výměnu dat mezi aplikacemi a pro publikování dokumentů.

YAML – Představuje formát pro serializaci struktorovaných dat. Nejčastěji se používá pro konfigurační soubory.

JSON – Textová prezentace strukturovaných dat bez schématu. Slouží pro přesnost dat, která mohou být organizována v polích nebo objektech.

HTTP – Protokol sloužící pro přenos souborů mezi serverem a klientem.

TCP – Protokol zajišťující navázaní a ukončení spojení mezi dvěma aplikacemi. Zároveň protokol garantuje spolehlivé doručování a doručování ve správném pořadí.

UPD - Protokol, který nezajišťuje spolehlivost ani pořadí přicházejích data. Zároveň ani nenavazuje spojení mezi aplikacemi, pouze se do sítě vyšlou nezávislé diagramy

SMTP server – Aplikace jejímž primárním účelem je přenos e-mailů.

TLS – Bezpečností protokol navržený k usnadnění ochrany soukromí a dat při komunikaci přes internet. Využívá se pro šifrování komunikace mezi webovými aplikacemi a servery. Dále se využívá pro šifrování komunikace emailů nebo jiných zpráv.

AJAX – Představuje asynchronní JavaScript pomocí, kterého lze aktualizovat data bez nutnosti znovunačtení stránky.

1. Úvod

Při výběru tématu bakalářské práce jsem hledal téma, které by mi bylo osobně blízké a kde bych mohl využít již nabité zkušenosti z minulosti. Z toho důvodu jsem si vybral jako téma vytvoření informačního systému hokejbalových hráčů, jelikož jsem se tomuto sportu věnoval během svého dětství a při vývoji samostatné aplikace mohu využít zkušenosti, které jsem získal během vývoje webových aplikací.

Webové aplikace v dnešní době patří k neodmyslitelné součásti našeho života. Na rozdíl od ostatních aplikací se nemusí instalovat a jsou dostupný ze všech zařízení. Pro jejich vývoj existují ucelené prostředky a definované jednotlivé postupy, které vysledný vývoj ulehčují. Informační systém jako takový má ulehčit práci pro evidenci dat a následné úpravy.

Cílem práce je vytvořit webovou aplikaci, ve kterém bude možnost vyhledat hráče a jejich týmy. Další důležitá část aplikace je následné upravování a vytváření nových dat, které bude možné po přihlášení do aplikace s určitou rolí. Výsledná aplikace bude dostupná online na všech zařízeních.

2. Cíl práce a metodika

Cílem bakalářské práce je vytvořit funkční informační systém hokejbalových hráčů. Systém bude vytvořen pomocí html,php,css a databáze, kde budou uložena jednotlivá data. Dále k otestovaní správné funkčnosti systému budou využity automatické testy, které budou vytvořeny v programu Pycharm s využitím robot framework.

Dle doporučených metod softwarového inženýrství provedu analýzu uživatelských požadavků, návrh designu, implementaci a řádné otestování informačního systému pomocí automatických testů.

3. Teoretická východiska.

1.1 Metodiky vývoje softwaru

Pro efektivní fungování týmu je potřeba správně zvolená metodika, která dokáže fungování celého projektu zlepšit. Ovšem špatně zvolená metodika dokáže fungování celého týmu klidně zhoršit. Proto je vždy důležité na začátku projít jednotlivé silné i slabé stránky jednotlivých metodik a podle toho vybrat tu, která bude celému týmu vyhovovat nejvíce.

1.1.1 Rigorózní metodiky

Jedná se o největší skupinu metodik, které se zabývají podrobným popisem procesu vývoje softwaru. Kladen je hlavně důraz na objemnou dokumentaci. Celkový vývoj softwaru je pak jasně definovaný proces, od kterého se nelze odchýlit. (1)

1.1.1.1 Vodopádový model

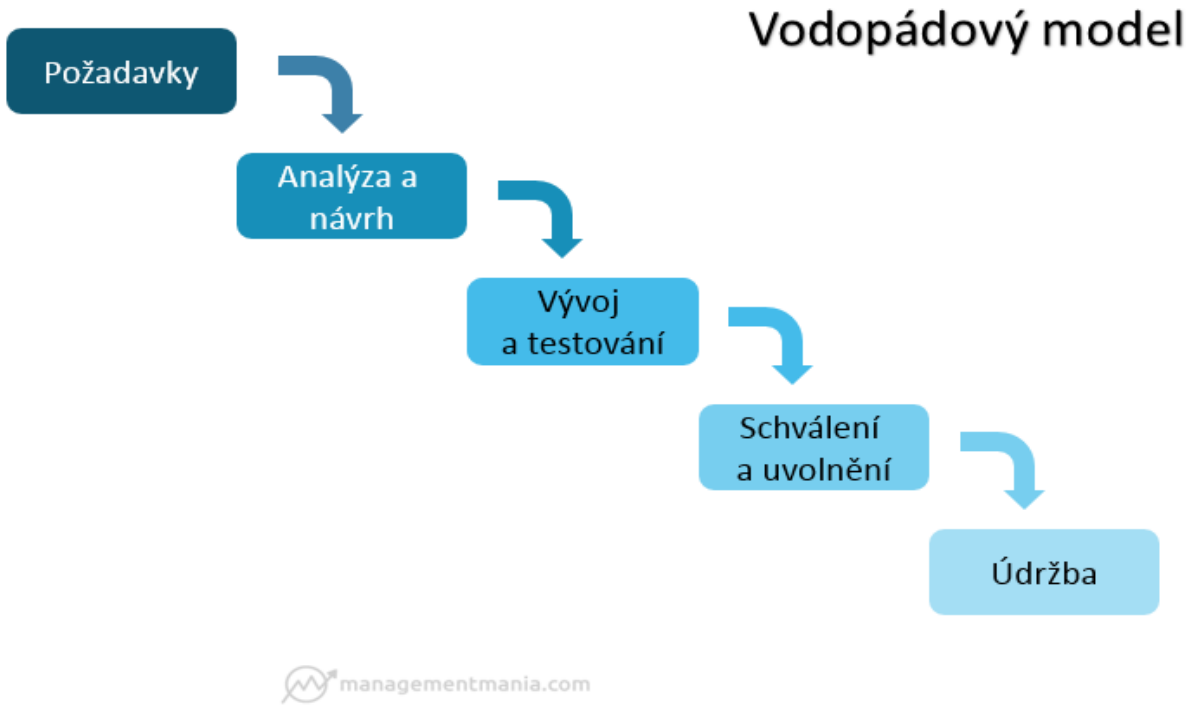
Model vodopádu byl prvním zavedeným procesním modelem. Model se skládá z postupných fázi (požadavky, analýza a návrh, implementace, verifikace a údržba) a každá z těchto fází je zaměřena na odlišené cíle. Tyto fáze jsou navzájem kaskádovité a další fáze je zahájena až poté, co je 100 % dokončena fáze předchozí. Hlavním předpokladem vodopádového modelu je, že do jednou ukončené fáze vývoje se nebudeme muset nikdy znovu vracet. Proto obvykle neexistuje ani žádný process, jak se vrátit zpět a upravit projekt nebo směr. Kvůli nemožnosti vracet se zpět je nejdůležitější částí projektu fáze analýzy a návrhu, která dokáže předejít případným kritickým chybám, které by mohly způsobit i nedokončení celého projektu, či předejít větší ztrátě peněz. Pokud se kritická chyba objeví v začátcích projektu, tak je možné projekt ještě poupravit, ovšem v tomto případě musí celý process začít úplně od začátku a to může stát firmu nemalé finanční prostředky. Tento model nejvíce najde uplatnění u menších projektů, kde jsou požadavky velmi dobře pochopeny. (2)

1.1.1.1.1 Silné stránky vodopádového modelu

Všechny procesy a výsledky jsou dobře zdokumentovány, díky tomu je velmi jednoduché pro nově příchozí lidi do projektu pochopit vyvíjenou aplikaci. Také lze nastavit termín pro každou fázi vývoje. (2)

1.1.1.1.2 Slabé stránky

Model neumožňuje mnoho sebereflexe nebo revize. Pokud je aplikace ve fázi testování je velmi obtížné vrátit se zpět a změnit něco, co nebylo ve fázi analýzy dobře zdokumentováno. Je zde také vysoká míra rizika a nejistoty a až do konce životního cyklu se nevyrábí žádný funkční software. Jakoukoliv úpravou rozsahu během životního cyklu může být projekt ukončen a v podstatě nelze vyhovět měnícím se požadavků.(2)



Obrázek 1 - Vodopádový model

Zdroj : https://managementmania.com/cs/vodopadovy-model-waterfall-model

1.1.2 Agilní metodiky

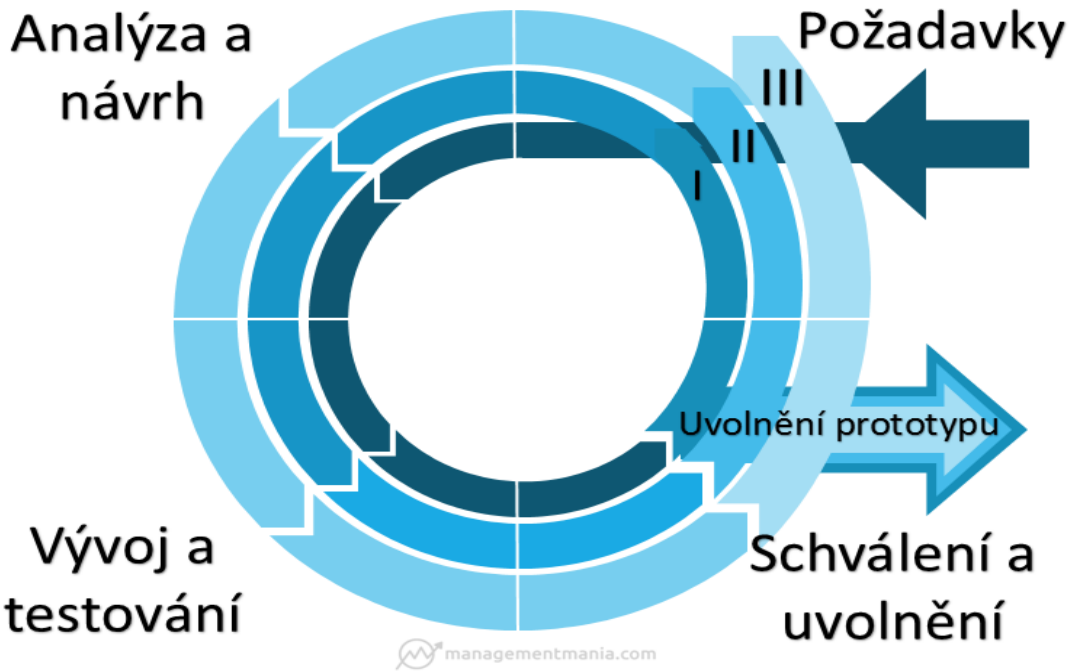
V roce 2001 formulovala skupina vývojářů principy agilních metodik a následně napsali Manifest agilního programování, který definuje 12 prvků. (3)

1. Naší nejvyšší prioritou je vyhovět zákazníkovi časným a průběžným dodáváním hodnotného softwaru. (3)

2. Vítáme změny v požadavcích, a to i v pozdějších fázích vývoje.
   Agilní procesy podporují změny vedoucí ke zvýšení konkurenceschopnosti zákazníka. (3)
3. Dodáváme fungující software v intervalech týdnů až měsíců s preferencí kratší periody. (3)
4. Lidé z byznysu a vývoje musí spolupracovat denně po celou dobu projektu. (3)
5. Budujeme projekty kolem motivovaných jednotlivců. Vytváříme jim prostředí, podporujeme jejich potřeby a důvěřujeme, že odvedou dobrou práci. (3)
6. Nejúčinnějším a nejefektivnějším způsobem sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace. (3)
7. Hlavním měřítkem pokroku je fungující software. (3)
8. Agilní procesy podporují udržitelný rozvoj. Sponzoři, vývojáři i uživatelé by měli být schopni udržet stálé tempo trvale. (3)
9. Agilitu zvyšuje neustálá pozornost věnovaná technické výjimečnosti a dobrému designu. (3)
10. Jednoduchost umění maximalizovat množství nevykonané práce je klíčová. (3)
11. Nejlepší architektury, požadavky a návrhy vzejdou ze samo-organizujících se týmů. (3)
12. Tým se pravidelně zamýšlí nad tím, jak se stát efektivnějším, a následně koriguje a přizpůsobuje své chování a zvyklosti. (3)

Agilní metodiky si na rozdíl od rigorózních metodik zakládají na průběžné komunikaci se zákazníkem a průběžným dodáváním menších částí aplikací. Díky tomuto lze do aplikace vždy zasáhnout a provést změnu požadovanou od zákazníka či přidat nový požadek. (1)

Tato flexibilita má zásluhu na tom, že jsou agilní metodiky velmi populární, a i velké korporáty jako je například Česká spořitelna se na ní rozhodly přemigrovat.



Obrázek 2 - Agilní metodika

Zdroj: https://managementmania.com/cs/agilni-projektove-rizeni

1.1.2.1 Scrum

Scrum je v dnešní době nejpoužívanější metodikou agilního řízení. Klíčovou roli v této metodice hraje komunikace mezi týmy, což v dnešní době, kdy je většina týmu na vzdáleném připojení může být složitější. Hlavnímy principy, na které se scrum metodika zaměřuje a spoléhá je transparentost, monitorování a neustálá přizpůsobivost. (4)

1.1.2.1.1 Role

Tým ve scrum metodice se skládá ze 3 rolích. Každá tato role má určité úkoly, které pro správný chod celého týmu musí plnit. (5)

Vlastník produktu funguje něco jako hlas zákazníka a na základě zákazníkových požadavků určuje priority pro tým. Dále se zaručuje zákazníkovi, že prioritní požadavky budou doručeny včas a určuje rozvrh doručení. Mezi další důležité úkoly vlastníka produktu patří shánění financí pro svůj tým. V případě nedostatku financí musí určit, které požadavky budou vynechány, aby se za přístupné finance udělaly jen nejdůležitější požadavky, které mají pro zákazníka největší hodnotu. (5)

Vývojový tým se skládá ze všech, kteří se na vývoji dané aplikace podílejí. Mezi členy týmu se většinou řadí vývojáři, analytici a testeři. Žádná role zde nemá větší váhu,

jelikož je každá tato role nesmírně důležitá pro správné fungování vyvíjené aplikace. Základem správného fungování týmu je vzájemná ochota si pomoct. Díky vzájemné pomoci se může urychlit proces vývoje některých nových funkčností nebo si členové mohou prohloubit své znalosti aplikací. (5)

Scrum master dohlíží na to, aby tým pochopil správné fugování scrum metodiky. Jeho hlavní náplní je odstranění překážek uvnitř týmu a snaží se navést tým na správnou cestu jakým by se měl ubírat. Dále se snaží tým povzbuzovat k lepším výkonům a domlouvá celotýmové schůzky. (5)

1.1.2.2 Události

Sprint trvá většinou 2 týdny a zahrnuje všechny níže zmíněné události. (5)

Denní scrum celotýmové ranní setkání, které se koná každý den sprintu a nemělo by zabrat víc jak 15 minut. Na tomto setkání odpovídá každý člen týmu na tři otázky. (5)

1. Co dělal včera.
2. Co bude dělat dnes.
3. Co ho blokuje.

Po zodpovězených těchto otázek má každý přehled, kdo na čem pracuje a případně kdo s čím potřebuje pomoct.

Plánování sprintu provádí vždy celý tým na začátku nového sprintu. Pro správné odhadnutí náročnosti jednotlivých požadavků se využívají story pointy, které jsou založeny na Fibonacciho posloupnosti. Na základě předchozích sprintů je možno odhadnout kolik je tým schopen story pointů splnit a podle toho, pak vlastník produktu ví, kolik story pointů může do dalšího sprintu zařadit. Vybírání jednotlivých vypsaných požadavků je pak na vlastníkovi produktu, který určuje prioritu, a vývojářů, kteří předpovídají kolik z toho budou schopni dodat včas. (5)

Zhodnocení sprintu se koná v poslední den sprintu. Umožňuje scrum týmu předvést zakazníkům nově vydané funkce aplikace. Kromě kontroly jednotlivých funkcí je zde i zpětná vazba od zákazníka, která může pomoct týmu v budoucích sprintech. (5)

Retrospektiva je poslední událostí sprintu. Tým zde píše klady a zápory posledního sprintu. Na základě záporů pak tým vyhodnocuje překážky ve sprintu a snaží se je odstranit. Po vymyšlení, jak by se daná překážka dala odstranit, je delegovaná na člena týmu, aby se daný problém zaměřil a pokusil se ho odstranit. (5)

1.2 Fáze vývoje softwaru

1.2.1 Plánování

Na začátku plánování určuje zákazník požadavky, které funkce by aplikace měla obsahovat. V této fázi určuje vedoucí projektu, kolik bude potřeba finančních prostředků. (6)

1.2.2 Analýza a design

Analýza se zaměřuje na jednotlivé požadavky zákazníka a zdali je možné dané požadavky realizovat. U realizovatelných požadavků musí poté analytik přesně specifikovat jednotlivé kroky, které se mají dít pro určitou funkci. Dále je zde zahrnut samotný návrh jak má aplikace vypadat a jak mají vypadat interaktivní prvky. (6)

1.2.3 Vývoj

Tato část je samostatné psaní programu. Menší projekt je schopen napsat jeden vývojář, s vyšší komplexitou projektou se práce rozděluje mezi větší skupinu vývojářů. Při vytváření kódu je potřeba dodržovat zásady týmu, aby byl napsaný kód pro ostatní co nejčitelnější. Důležité je taky napsání J-Unit testů, které se pouštějí při každém nasazení aplikace. (6)

1.2.4 Testování

1.2.4.1 Manuální testování

Výchází z předem napsaných testovacích scénářů, které obsahují jednotlivé kroky, jak se má aplikace chovat. Tyto kroky musí být splněny. Při nesplnění jakéhokoliv kroku je ihned potřeba informovat tým o chybě a test označit jako neúspěšný. Scénář většinou bývá napsán samotnými testery a pokud to kapacita dovoluje je lepší, když ho testuje jiný tester než ten, jenž ho psal. (7)

1.2.4.2 Automatizované testování

Funguje jako pomoc při testování. Nejčastěji se automatické testování používá pro základní funkce programu jako je přihlašování, správné vyhledávání dat nebo správné zobrazování dat. Účelem je ušetření času testerům, kteří se poté mohou více věnovat komplexnějším funkcím. (7)

1.2.5 Provoz a údržba

Po úspěšném nasazení aplikace na produkční prostředí je potřeba zajistit funkční podporu, na kterou se v případě potíží s aplikací zašle zpětná vazba. Je velmi obvyklé, že

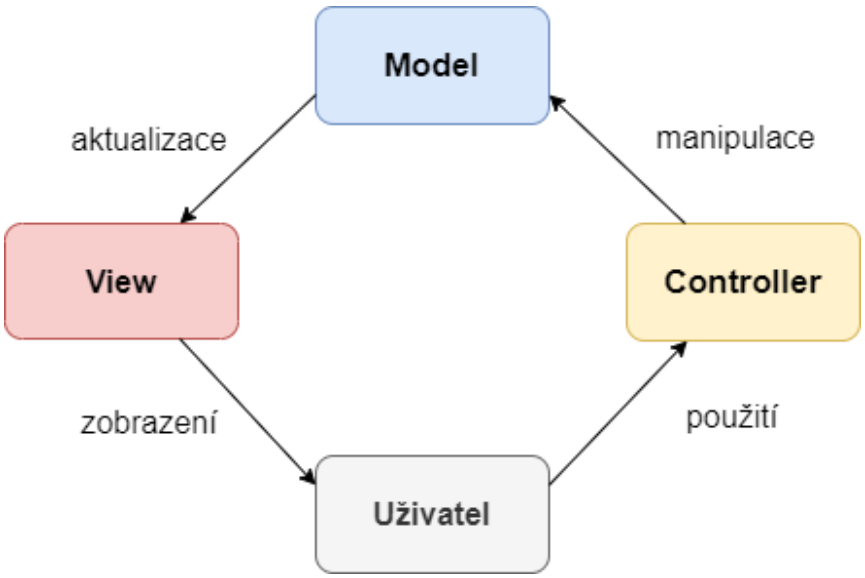
některé chyby objeví až samotný zákazník, tudíž je funkční podpora nesmírně důležitá pro každou aplikaci. Po obdržení jakéhokoliv problému tým provede analýzu a snaží se chybu co nejdříve odstranit. Většinou je obvyklé zákazníkovi poskytnout alternativní řešení, aby mohl pokračovat v používání aplikace tentýž den. (7)

1.3 Architektonické vzory

Architektonický vzor si lze představit jako obecné a opakovatelně použitelné řešení běžně se vyskytujícího problému. Architektonické vzory mají určitou podobu se vzory návrhovými, ale liší se širším rozsahem.

1.3.1 MVC

Model-View-Controller je složený ze tří vzájemně propojených částí. Hlavní cíl tohoto vzoru je oddělení logiky aplikace od zbytku uživatelského rozhraní. Řeší tedy problém tzv. "špagetového kódu", kdy jsou v jednom souboru smíchané dohromady databázové dotazy, logika aplikace a výpis. Kód se samozřejmě špatně udržuje, natož rozšiřuje. Je špatně highlitovaný, protože si s ním vývojové prostředí neví rady. HTML není správně naformátováno a ztrácíme se v jeho stromové struktuře. Cílem MVC modelu je, aby zdrojový kód s logikou vypadal jako zdrojový kód (např. PHP, java) a výstup vypadal jako HTML stránka s co nejmenší příměsí dalšího kódu. Pokud jsou tyto kroky splněny, je údržba kódu velmi jednoduchá a jednoduše se rozšiřuje o další funkce. (8)



Obrázek 3 - MVC model

1.3.1.1 Model

Slouží k získavání, úpravě a zakládání dat v databází. Na základně událostí z **controlleru** provádí určité činnosti. (8)

1.3.1.2 View

Část vykreslující přijatá data z **modelu** do podoby vyhovující zákazníkovi (nejčastěji formou HTML). Slouží také jako interaktivní část se zákazníkem, která zachytává události od uživatele (například kliknutí myší na určité tlačítko, které je vykresleno), tyto události jsou následně předávány do **controlleru.** (8)

1.3.1.3 Controller

Slouží jako propojení mezi **model** a **view.** Na základě zpracovaných událostí z **view** od uživatele volá příslušné funkce **modelu**. Podle příslušných akcí z **modelu** následně vybírá vhodné **view** pro zobrazení příchozích dat. (8)

1.4 API

API je rozhraní, které slouží pro programování aplikací a umožňuje interakci, přístup a výměnu dat nebo funkcí mezi dvě systémy. Druhů API je několik, ale nejběžnější jsou webové API. Uživatelé API funkce využívají denně, nejznámější je API, které umožňuje uživatelům se přihlásit pomocí jejich Google účtu do různých aplikací. (11)

1.4.1 REST

REST bylo představeno v roce 2000 v disertační práci Roye Fieldinga. Tato architektura umožnuje doručovat data ve více formátech, jako je prostý text, HTML, XML, YAML a JSON. Nejpoužívanějším formátem je již zmíněný JSON, jelikož lze jednoduše analyzovat a je to lehký formát pro výměnu dat. REST jako takový je schopen komunikovat pouze HTTP protokolem, ovšem celkově je výkonově lepší než SOAP řešení. (9)

1.4.1.1 Základní metody pro práci s daty

GET metoda sloužící k načtení dat ze systému.

POST metoda sloužící k vytvoření nových dat v systému.

PUT metoda sloužící pro aktualizaci již existujích dat. Metoda jako taková je velmi podobná metodě **POST,** s tím rozdílem, že se volá konkrétní URI konkrétního zdroje, který chceme změnit a v těle se předává nová hodnota.

DELETE metoda sloužící k smázání již existujích záznamů.

[{"id":"2","first_name":"Dan","last_name":"Novák","birth_date":"2001-06-01"},
{"id":"3","first_name":"Tomáš","last_name":"Novotný","birth_date":"1998-03-02"}]

Obrázek 4 - Příklad JSON zprávy

1.4.2 SOAP

Simple Object Access Protocol je protokol pro výměnu dat v distribuovaných systémech založený na formátu XML. SOAP je schopno pracovat s jakýmkoliv protokolem aplikační vrstvy jako je HTTP, SMTP, TCP nebo UDP. Zabezpečení, autorizace a zpracování chyb jsou přímo integrovány do protokolu. Struktura samotného protokolu se řídí formálním a standardizovaným přístupem, který určuje, jak kódovat soubory XML vrácené rozhraním API. Samotný XML soubor obsahuje následující části. (10)

1. **Envelop (povinné) –** Počáteční a koncová značka zprávy
2. **Header (volitelné) –** Obsahuje volitelné atributy zprávy
3. **Body (volitelné) –** Obsahuje data, která server přenáší do přijímače
4. **Alert (volitelné)** – Přenáší informace o chybách, ke kterým došlo během zpracování zprávy

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:c="http://www.acmeOrders.com/OrderService"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Body>
        <c:OrderMessage>
            <localElement>
                <FirstName>John</FirstName>
                <LastName>Smith</LastName>
                <Street>High Street</Street>
                <City>London</City>
                <ZipCode>W1A1AA</ZipCode>
                <PartNumber>ABC1234</PartNumber>
                <Quantity>1</Quantity>
            </localElement>
        </c:OrderMessage>
    </soap:Body>
</soap:Envelope>
```

Obrázek 5 - Příklad SOAP zprávy

Zdroj: https://cutt.ly/FkY0BGP

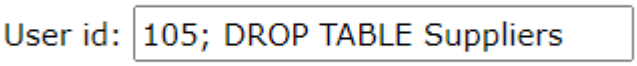
1.4.3 Srovnání REST a SOAP služeb

REST architektura nabila oblibu díky své jednoduchosti, škálovatenosti, flexibilitě a lepšímu výkonu. V roce 2017 bylo zaznamenáno 82,2% (11) volání právě přes tuto architekturu. Nejčastěji se používá jako veřejné API pro webové služby, sociální sítě a mobilní služby. SOAP díky svému vyššímu zabezpečení oproti REST architektuře stále nachází své uplatnění. Tento protokol je aktivně využíván v bankách, kde je vysoké zabezpečení velmi důležité a v platebních branách. SOAP API od firmy PayPal patří mezi nejznámější. Pomocí toho rozhraní mohou uživatelé na různých webových stránkách zaplatit bezpečně částku online, přijímat platby nebo provádět další různé akce související s PayPal. (10)

1.5 SQL Injection

SQL Injection je nejjednodušší způsob jak lze napadnout databázovou vrstvu aplikace. Přes neošetřené vstupy může útočník jednoduše zadat SQL příkaz, který se pak bez vašeho vědomí pustí v databázi. Následky takového útoku mohou být nedozírné. Utočník může například smazat libovolnou tabulku v databázi nebo se přihlásit na libovolný účet. (12)

V dnešní době je většina webů na tento typ útoku připravena, ale i tak se můžou objevit webové stránky, které lze takto jednoduše napadnout a to hlavně kvůli neznalosti programátora dané problematiky.



Obrázek 6 - Příklad SQL Injection

Zdroj: https://www.w3schools.com/sql/sql_injection.asp

4. Vlastní práce

1.6 Cíle aplikace

Cílem práce je vytvoření informačního systému hokejbalových hráčů, na který by měla možnost přístupu a základního vyhledávání i široká veřejnost. Systém by měl být na celorepublikové úrovni podobně jako informační systém FAČR (facr). V současné době sice existuje jako informační systém stránka pro správu klubu (is.cmshb.cz), ovšem ta není přístupná pro širokou veřejnost a funguje jen jako systém pro určitý klub.

Systém by měl obsahovat možnost vyhledávání hráčů a týmu pro širokou věřejnost. Dále by v systému měla být správa uživatelů, kteří budou moct již zadaná data upravovat a případně vytvářet nová. V neposlední řadě by zde měla být možnost zadávání přestupů mezi jednotlivými týmy.

1.7 Využité technologie

Pro samostaný výběr technologií vhodných pro tvorbu informačního systému hokejbalových hráčů a následné automatické testování, jsem využil předešlé zkušenosti z vyvíjení aplikací.

1.7.1 Technologie pro tvorbu aplikace

1.7.1.1 Bootstrap

Bootstrap je v dnešní době jeden z nepopulárnějších CSS frameworků vytvořený společností Twitter. Framework je velmi oblíbený hlavně díky grid systému, díky kterému je možné během chvilky vytvořit responzivní webou stránku. (13)

1.7.1.2 jQuery

jQuery je velmi oblíbený framework JavaScriptu jehož cílem je usnadnění práce vyvojářům. Dále umožňuje vývojářům využívat AJAX metody. (14, s. 517)

1.7.1.3 PHP

PHP je skriptovací jazyk na straně serveru, který byl speciálně vytvořen pro web. Jedná se o projekt s otevřeným zdrojovým kódem, tudíž je možné ho libovolně měnit a distribuovat ho. (14, s. 30)

1.7.1.4 MySQL

Nejoblíbenější databází s otevřeným kódem je MySQL. Jedná se o velmi rychlý systém pro správu relačních databází. (14. s. 31). Komunikace s tímto systémem probíhá za pomocí jazyka SQL.

1.7.2 Technologie pro automatické testy

1.7.2.1 Robot Framework

Robot Framework je open source software využívající testování řízené pomocí klíčových slov. Jádro frameworku je založeno především na programovacím jazyce Python. (15)

K frameworku existuje řada knihoven a nástrojů, většina z nich je open source, a jsou založeny jako samotný robot framework na jazyce Python nebo na jazyce Java.

1.7.2.2 Python

Python je vysokoúrovňový skriptovací jazyk, který je možno využít v jakémkoliv prostředí (16). V dnešní době patří mezi nejvíce používané programovací jazyky a díky své jednoduché syntaxi je vhodný i pro úplné začátečníky.

1.8 Use case spefikace

1.8.1 Příhlášení

Popis: Use case umožňuje přihlášení uživatele.

Aktéři: nepřihlášený uživatel, systém.

Podmínky pro spuštění: Uživatel není přihlášený a je na příslušné obrazovce.

Základní tok:

1. Systém vygeneruje formulář pro přihlášení.
2. Uživatel vyplní povinná pole, uživatelské jméno a heslo. Po vyplnění údajů stiskne tlačítko „Přihlásit“.
3. Systém zvaliduje zadaná data.
4. Systém přihlásí uživatele.

Alternativní tok: Při zadání nevalidních dat systém uživatele upozorní a zamítne přihlášení.

Podmínky pro dokončení: Uživatel bude přihlášen a přesměrován na stránku pro vyhledávání hráčů.

1.8.2 Přihlášení jednorázovým heslem

Popis: Use case umožňuje přihlášení uživatele pomocí jednorázového hesla.

Aktéři: nepřihlášený uživatel, systém.

Podmínky pro spuštění: Uživatel se přihlašuje pomocí jednorázového hesla.

Základní tok:

1. Systém vygeneruje formulář pro přihlášení.
2. Uživatel vyplní povinná pole, uživatelské jméno a heslo. Po vyplnění údajů stiskne tlačítko „Přihlásit“.
3. Systém zvaliduje data.
4. Systém zkontroluje jestli se uživatel přihlašuje pomocí jednorázového hesla.
5. Systém přesměruje uživatele na stránku pro změnu hesla.

Alternativní tok 1: Při zadání nevalidních dat systém uživatele upozorní a zamítne přihlášení.

Alternativní tok 2: Pokud se uživatel nepřihlašuje pomocí jednorázového hesla, systém uživatele rovnou přihlásí.

Podmínky pro dokončení: Uživatel bude přesměrován na stránku pro změnu hesla.

1.8.3 Změna hesla

Popis: Use case umožňuje uživateli změnu hesla.

Aktéři: přihlášený uživatel, systém.

Podmínky pro spuštění: Uživatel je přihlášený a nachází se na příslušné obrazovce.

Základní tok:

1. Systém vygeneruje formulář pro změnu hesla.
2. Uživatel vyplní povinná pole. Po vyplnění polí klikne na tlačítko „Změnit heslo“.
3. Systém zvaliduje data.
4. Systém uživatele odhlásí.
5. Systém uživatele přesměruje na stránku pro přihlášení s informativní hláškou o úspěšném změnění hesla.

Alternativní tok: Při zadání nevalidních dat systém uživatele upozorní a zamítne změnu hesla.

Podmínky pro dokončení: Uživatel bude odhlášen a přesměrován na stránku pro přihlášení s informativní hláškou.

1.8.4 Zapomenuté heslo

Popis: Use case umožňuje vygenerování nového jednorázového hesla

Aktéři: nepřihlášený uživatel, systém.

Podmínky pro spuštění: Uživatel je odhlášený a nachází se na příslušné stránce.

Základní tok:

1. Systém vygeneruje formulář pro vygenerování jednorázového hesla.
2. Uživatel vyplní uživatelské jméno a klikne na tlačítko „Zaslat nové heslo“.
3. Systém zkontroluje existenci uživatelského jména v databázi.
4. Systém vygeneruje nové jednorázové heslo, které bude následně zasláno uživateli na jeho emailovou adresu uloženou v databázi.
5. Systém uživatele přesměruje na stránku pro přihlášení s informativní hlášku o zaslání jednorázového hesla na email.

Alternativní tok: Uživatel zadá neexistující uživatelské jméno, systém uživatele upozorní o neexistenci uživatelského jména a zamítne odeslání jednorázového hesla.

Podmínky pro dokončení: Uživateli bude zasláno nové jednorázové heslo a bude přesměrován na stránku pro přihlášení s informativní hláškou o zaslání hesla.

1.8.5 Odhlášení

Popis: Use case umožňuje odhlášení uživatele.

Aktéři: přihlášený uživatel, systém.

Podmínky pro spuštění: Uživatel je přihlášený.

Základní tok:

1. Uživatel klikne na položku v menu „Odhlásit“.
2. Systém uživatele odhlásí a bude přesměrován na vyhledávání hráčů.

Podmínky pro dokončení: Uživatel bude odhlášen a přesměrován na stránku pro vyhledávání hráčů.

1.8.6 Detail uživatele

Popis: Use case umožňuje zobrazit detail uživatele.

Aktéři: přihlášený uživatel s rolí admin, systém.

Podmínky pro spuštění: V databázi existuje vytvořený uživatel.

Základní tok:

1. Uživatel přistoupí na stránku pro zobrazení detailu uživatel.
2. Systém vyplní jednotlivá pole s údaji uživatele.

Alternativní tok: Při neexistenci uživatele systém přesměruje uživatele na stránku pro vyhledávání uživatelů.

Alternativní tok: Pokud uživatel nebude mít roli admin bude přesměrován na stránku pro vyhledávání hráčů.

Podmínky pro dokončení: Bude zobrazen detail uživatele s údaji z databáze.

1.8.7 Vytvoření uživatel

Popis: Use case umožňuje vytvoření nového uživatele.

Aktéři: přihlášený uživatel s rolí admin, systém.

Podmínky pro spuštění: Přihlášený uživatel s rolí admin nacházející se na příslušné obrazovce.

Základní tok

1. Systém vygeneruje formulář pro vytvoření nového uživatele.
2. Uživatel vyplní všechna povinná pole a vybere roli. Po vyplnění stiskne tlačítko „Vytvořit“.
3. Systém zvaliduje data.
4. Systém zkontroluje jestli uživatel se zadaným emailem již neexistuje.
5. Systém vygeneruje nové přihlašovací jméno s jednorázovým heslem a tyto údaje zašle na zadaný email.
6. Systém uloží nového uživatele do databáze.
7. Systém přesměruje uživatele na stránku pro vytvoření nového uživatele s informativní hláškou o úspěšném založení.

Alternativní tok 1: Při zadání nevalidních dat systém uživatele upozorní a zamítne vytvoření nového uživatele.

Alternativní tok 2: Při zadání již existujícího emailu systém uživatele upozorní na existenci uživatele a zamítne vytvoření nového uživatele.

Podmínky pro dokončení: Bude založen nový uživatel v databázi a novému uživatelu se zašlou přihlašovací údaje pro přihlášení.

1.8.8 Editace uživatele

Popis: Use case umožňuje vytvoření nového uživatele.

Aktéři: přihlášený uživatel s rolí admin, systém.

Podmínky pro spuštění: Přihlášený uživatel s rolí admin nacházející se na příslušné obrazovce.

Základní tok

1. Systém vygeneruje formulář pro editaci uživatele.
2. Uživatel vyplní všechna povinná pole a vybere roli. Po vyplnění stiskne tlačítko „Vytvořit“.
3. Systém zvaliduje data.
4. Systém zkontroluje jestli byl email uživatele změněn a pokud ano tak zkontroluje jestli zadaný email již není v databázi.
5. Systém aktualizuje data uživatele v databázi.
6. Systém ukončí režim editace.

Alternativní tok 1: Při zadání nevalidních dat systém uživatele upozorní a zamítne editaci uživatele.

Alternativní tok 2: Při zadání již existujícího emailu systém uživatele upozorní na existenci uživatele a zamítne vytvoření nového uživatele.

Podmínky pro dokončení: Bude provedena aktualizace dat uživatele v databázi.

1.8.9 Vyhledávání hráčů, týmu, uživatelů

Popis: Use case umožňuje zobrazení seznamu hráčů, týmu nebo uživatelů

Aktéři: všichni uživatelé, systém.

Podmínky pro spuštění: Uživatel je na příslušné stránce pro vyhledávání.

Základní tok:

1. Uživatel vyplní filtr nebo nechá filtr prázdný. Následně klikne na tlačítko „Vyhledat“.

2. Systém vygeneruje na stránku tabulky s daty podle vyplněného filtru. V případě, že nebyla ve filtru vyplněná žádná pole zobrazí se všechna data co se nacházejí v databázi.

Alternativní tok: V databází nebudou nalezeny žádné záznamy podle filtru, systém zobrazí místo tabulky s daty, informaci o tom, že podle filtru nebyla nalezená žádná data.

Podmínky pro dokončení: Na stránku systém vygeneruje tabulku s nalezenými daty.

1.8.10 Detail hráče, týmu

Popis: Use case umožňuje zobrazení detailu hráče nebo týmu.

Aktéři: všichni uživatelé, systém.

Podmínky pro spuštění: V databázi existuje vytvořený tým nebo hráč.

Základní tok:

1. Uživatel přistoupí na stránku pro zobrazení detailu týmu nebo hráče.
2. Systém vyplní jednotlivá pole s údaji týmu nebo hráče.

Alternativní tok: Při neexistenci týmu nebo hráče systém přesměruje uživatele na stránku pro vyhledávání.

Podmínky pro dokončení: Bude zobrazen detail hráče nebo týmu.

1.8.11 Vytvoření hráče

Popis: Use case umožňuje vytvoření hráče.

Aktéři: přihlášený uživatel s rolí admin nebo editor, systém.

Podmínky pro spuštění: Přihlášený uživatel s rolí admin nebo editor, nacházející se na přislušné stránce.

Základní tok

1. Systém vygeneruje formulář pro vytvoření nového hráče.
2. Uživatel vyplní všechna povinná pole.
3. Systém zvaliduje data.
4. Systém zkontroluje jestli je rodné číslo validní.
5. Systém zkontroluje jestli hráč se zadaným rodným číslem již neexistuje.
6. Systém uloží nového hráče do databáze.
7. Systém přesměruje uživatele na stránku detailu nově založeného hráče.

Alternativní tok 1: Při zadání nevalidních dat systém uživatele upozorní a zamítne vytvoření nového hráče.

Alternativní tok 2: Při zadání již zaevidovaného rodného čísla, systém uživatele upozorní na existenci uživatele a zamítne vytvoření nového hráč.

Podmínky pro dokončení: Bude založen nový hráč v databázi a uživatel bude přesměrován na detail nového hráče.

1.8.12 Editace hráče

Popis: Use case umožňuje editaci hráče.

Aktéři: přihlášený uživatel s rolí admin nebo editor, systém.

Podmínky pro spuštění: Přihlášený uživatel s rolí admin nebo editor, nacházející se na příslušné stránce.

Základní tok

1. Systém vygeneruje formulář pro editaci hráče.
2. Uživatel vyplní všechna povinná pole.
3. Systém zvaliduje data.
4. Systém zkontroluje jestli zadané rodné číslo nemá jiný uživatel.
5. Systém provede aktualizaci dat hráče v databázi.
6. Systém ukončí režim editace a zobrazí informativní hlášku o úspěšném změnění dat.

Alternativní tok 1: Při zadání nevalidních dat systém uživatele upozorní a zamítne editaci hráč.

Alternativní tok 2: Při zadání nevalidních dat systém uživatele upozorní a zamítne editaci hráč.

Podmínky pro dokončení: Bude provedena aktualizace dat hráče v databázi.

1.8.13 Vytvoření týmu

Popis: Use case umožňuje vytvoření týmu.

Aktéři: přihlášený uživatel s rolí admin nebo editor, systém.

Podmínky pro spuštění: Přihlášený uživatel s rolí admin nebo editor, nacházející se na příslušné stránce.

Základní tok

1. Systém vygeneruje formulář pro vytvoření nového týmu.
2. Uživatel vyplní všechna povinná pole.
3. Systém zvaliduje data.
4. Systém uloží nový tým do databáze.
5. Systém přesměruje uživatele na stránku detailu nově založeného týmu s informativní hláškou o úspěšném založení.

Alternativní tok: Při zadání nevalidních dat systém uživatele upozorní a zamítne vytvoření nového týmu.

Podmínky pro dokončení: Bude založen nový tým v databázi a uživatel bude přesměrován na detail nového týmu.

1.8.14 Editace týmu

Popis: Use case umožňuje editaci týmu.

Aktéři: přihlášený uživatel s rolí admin nebo editor, systém.

Podmínky pro spuštění: Přihlášený uživatel s rolí admin nebo editor, nacházející se na příslušné stránce.

Základní tok

1. Systém vygeneruje formulář pro vytvoření nového týmu.
2. Uživatel vyplní všechna povinná pole.
3. Systém zvaliduje data.
4. Systém provede aktualizaci dat týmu v databázi.
5. Systém ukončí režim editace a zobrazí informativní hlášku o úspěšném změnění dat.

Alternativní tok: Při zadání nevalidních dat systém uživatele upozorní a zamítne editaci týmu.

Podmínky pro dokončení: Bude provedena aktualizace dat v databázi.

1.8.15 Přestup

Popis: Use case umožňuje přestup hráče.

Aktéři: přihlášený uživatel s rolí admin nebo editor, systém.

Podmínky pro spuštění: Přihlášený uživatel s rolí admin nebo editor, nacházející se na příslušné stránce.

Základní tok

1. Systém vygeneruje formulář pro přestup hráče.
2. Uživatel vyplní všechna povinná pole.
3. Systém zvaliduje data.
4. Systém zaeviduje přestup do databáze, změní současný tým hráče.
5. Systém přesměruje uživatele na stránku detailu nově založeného týmu.

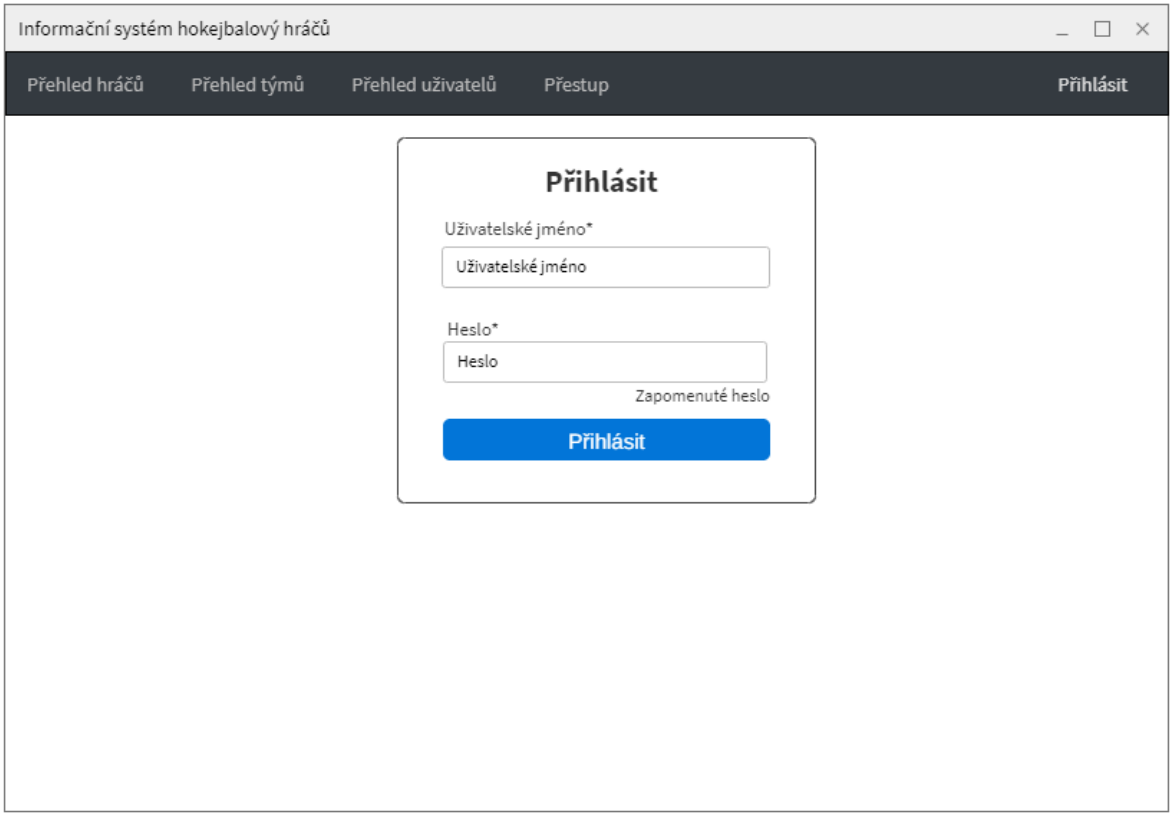
Alternativní tok: Při zadání nevalidních dat systém uživatele upozorní a zamítne přestup.

Podmínky pro dokončení: Bude založen nový záznam v databázi o přestupu a hráči se změní současný tým.

1.9 Drátěný model

Při vytváření drátěných modelů neboli wireframe modelů byly využity služby webové aplikace https://mockflow.com/. Hlavním důvodem tohoto výběru byla možnost vložení jednotlivých prvků z frameworku bootstrap, který byl vybrán pro vývoj samotné aplikace. Při vytváření wireframů byla vynechána obrazovka pro vytvoření týmu, jelikož je totožná s obrazovkou pro editaci týmu.

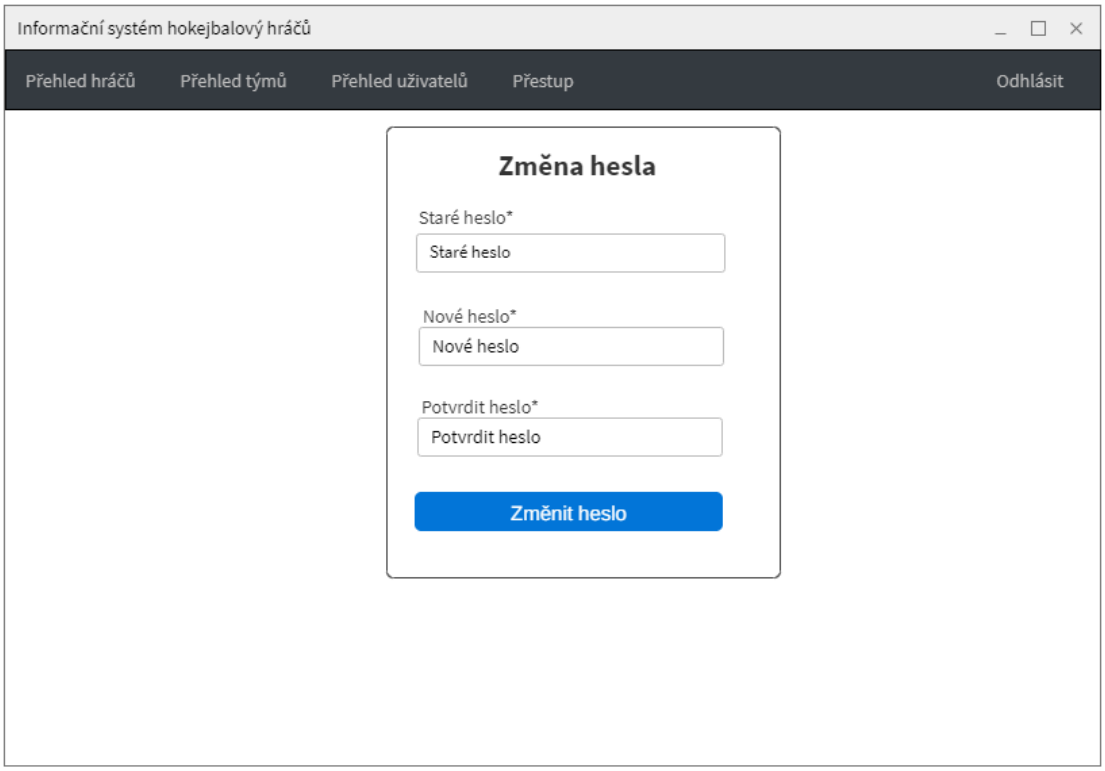
1.9.1 Přihlášení do aplikace



Obrázek 7 - Wireframe přihlášení

Pro přihlášení do aplikace je uživatel povinnen vyplnit pole uživatelské jméno a heslo. V případě zapomenutého hesla může uživatel využít proklik pod polem heslo, kterýho ho přesměruje na stránku pro zaslání nového hesla.

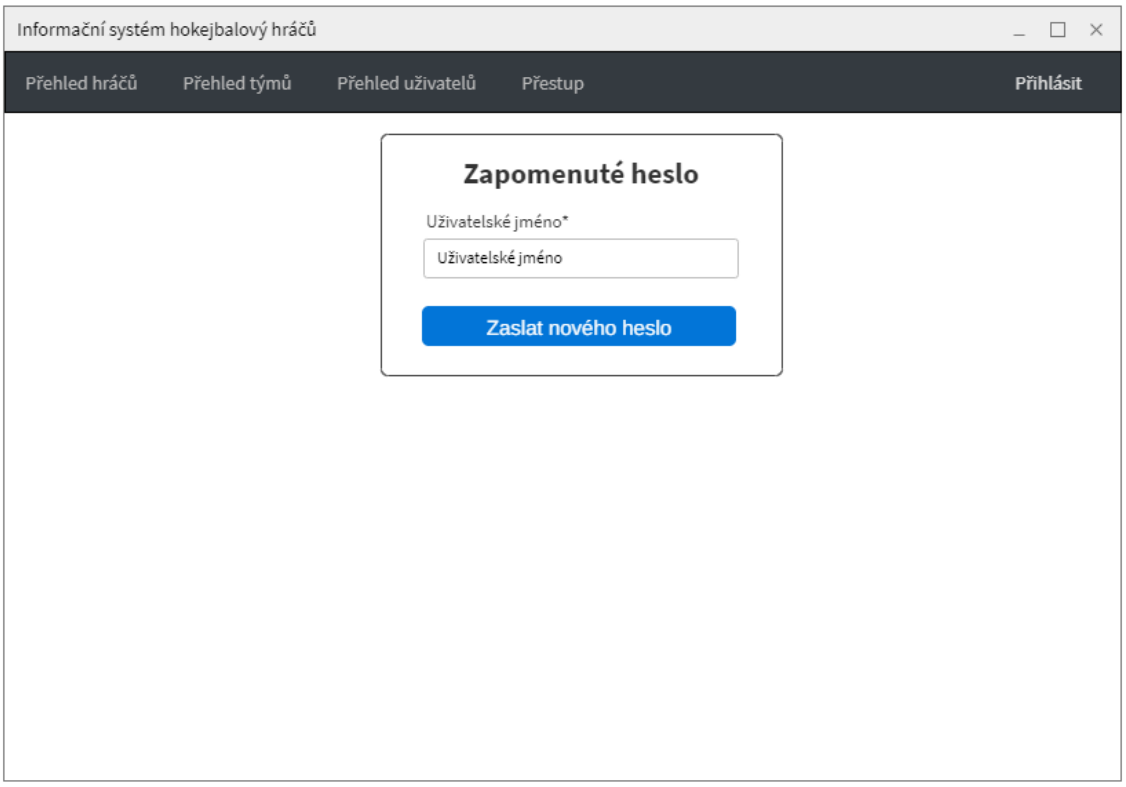
1.9.2 Změna hesla



Obrázek 8 - Wireframe pro změnu hesla

V případě nevyhovujícího současného hesla si ho uživatel může jednoduše změnit na obrazovce pro změnu hesla. V případě přihlášení pomocí jednorázového hesla se uživateli zobrazí pouze tato obrazovka, aby si uživatel nastavil heslo trvalé.

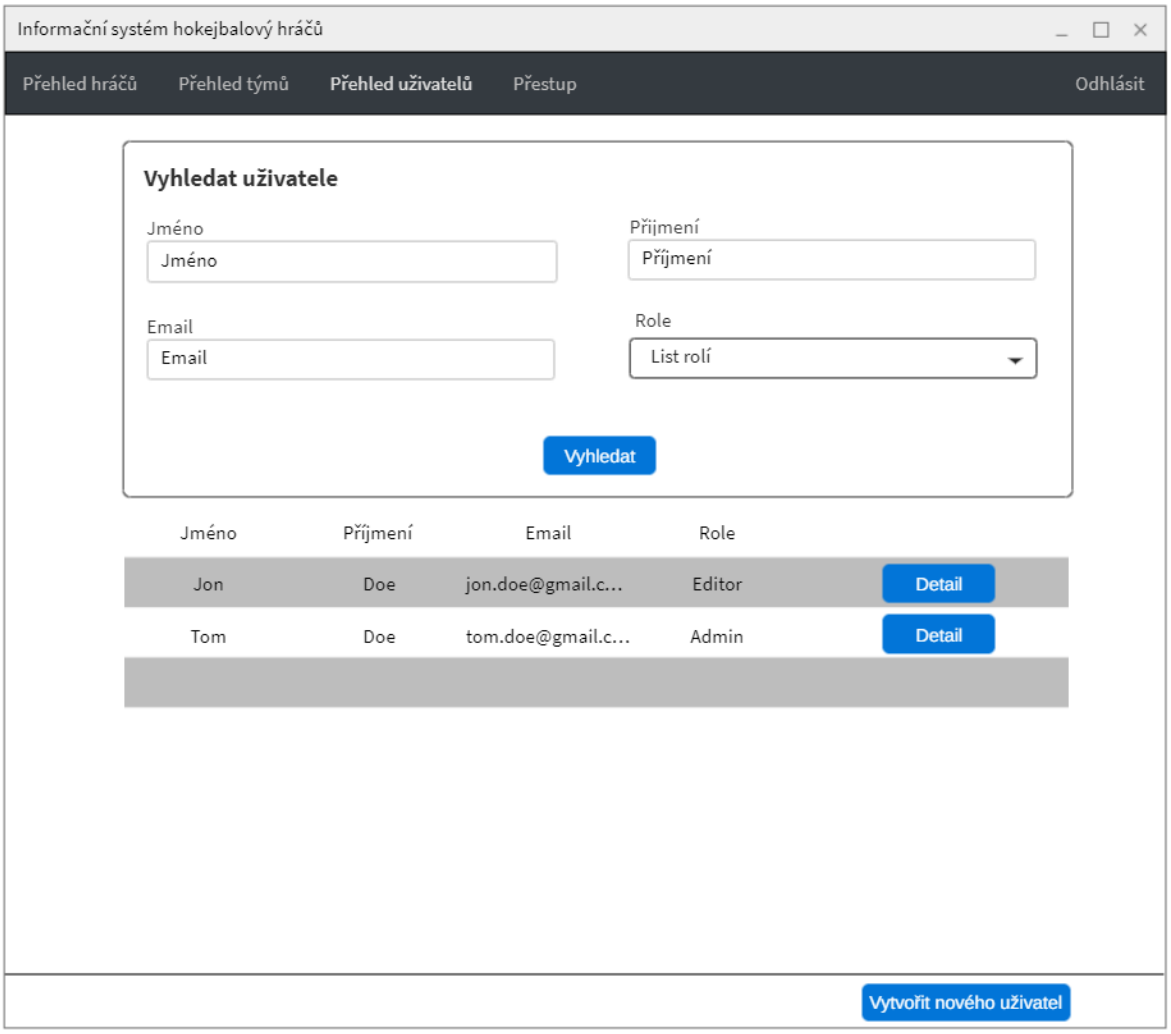
1.9.3 Zapomenuté heslo



Obrázek 9 - Wireframe pro zapomenuté heslo

V případě zapomenutého hesla si uživatel může jednoduše nechat zaslat nové jednorázové heslo za pomoci výše zobrazené obrazovky. Uživateli stačí vyplnit jeho uživatelské jméno a poté kliknout na tlačítko odeslat, systém poté zašle nové jednorázové heslo na email uživatele.

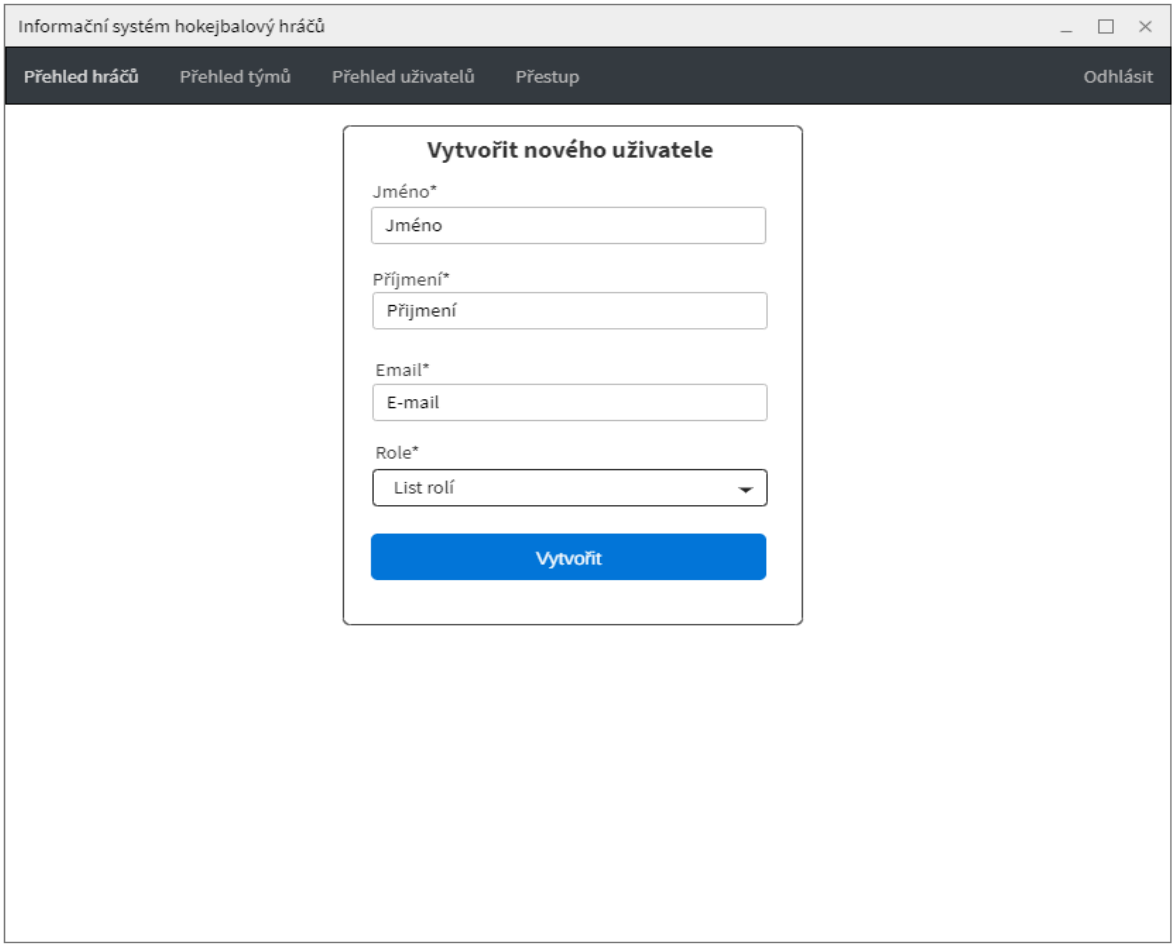
1.9.4 Vyhledání uživatele



Obrázek 10 - Wireframe pro vyhledání uživatele

Na stránku pro vyhledání uživatele má přístup pouze administrátor webové aplikace. V horní části obrazovky se nachází filtr pro vyhledávání konkrétního uživatele nebo uživatelů. V případě vyhledání dat se pod filtr vytvoří tabulka s příslušnými daty a tlačítkem na zobrazení detailu uživatele. Pokud se nevyhledají žádná data vypíše se místo tabulky informativní hláška. V dolní části obrazovky se nachází tlačítko „Vytvořit nového uživatele“, které je zobrazeno vždy jelikož na samostatnou stránku má přístup jen administrátor.

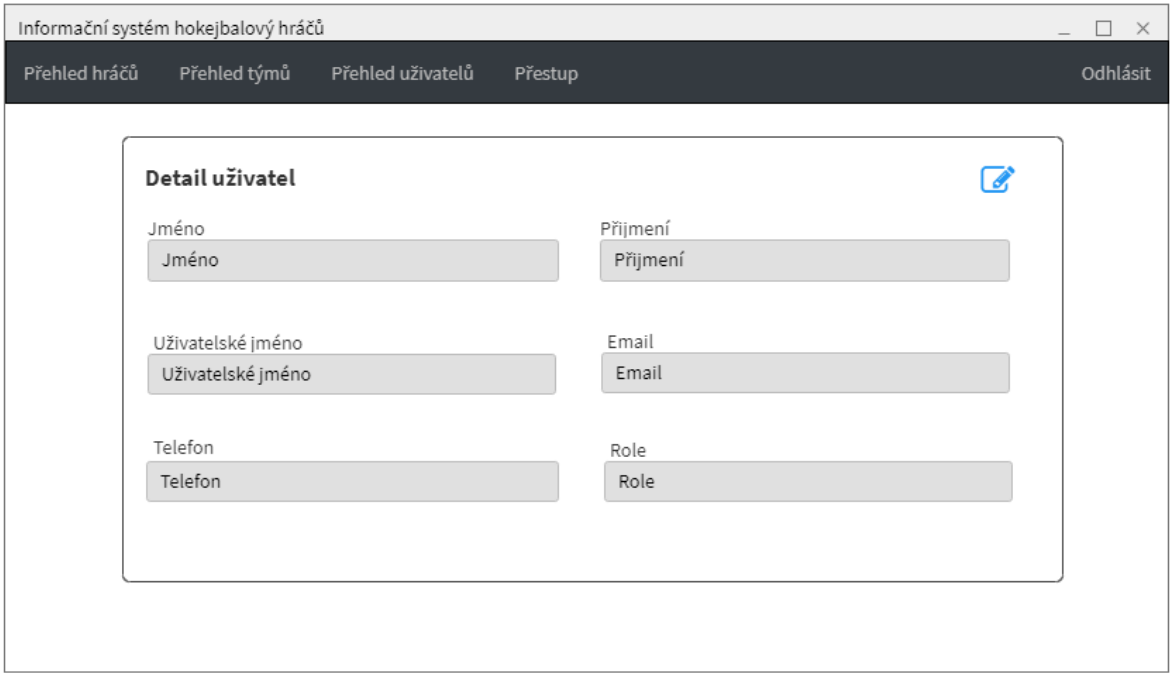
1.9.5 Vytvoření uživatele



Obrázek 11 - Wireframe pro vytvoření uživatele

Pro přístup na stránku na vytvoření nového uživatele je zapotřebí role admin. Při vytváření je potřeba vyplnit všechna povinná pole (jméno, příjmeni, email) a vybrat roli pro nového uživatele.

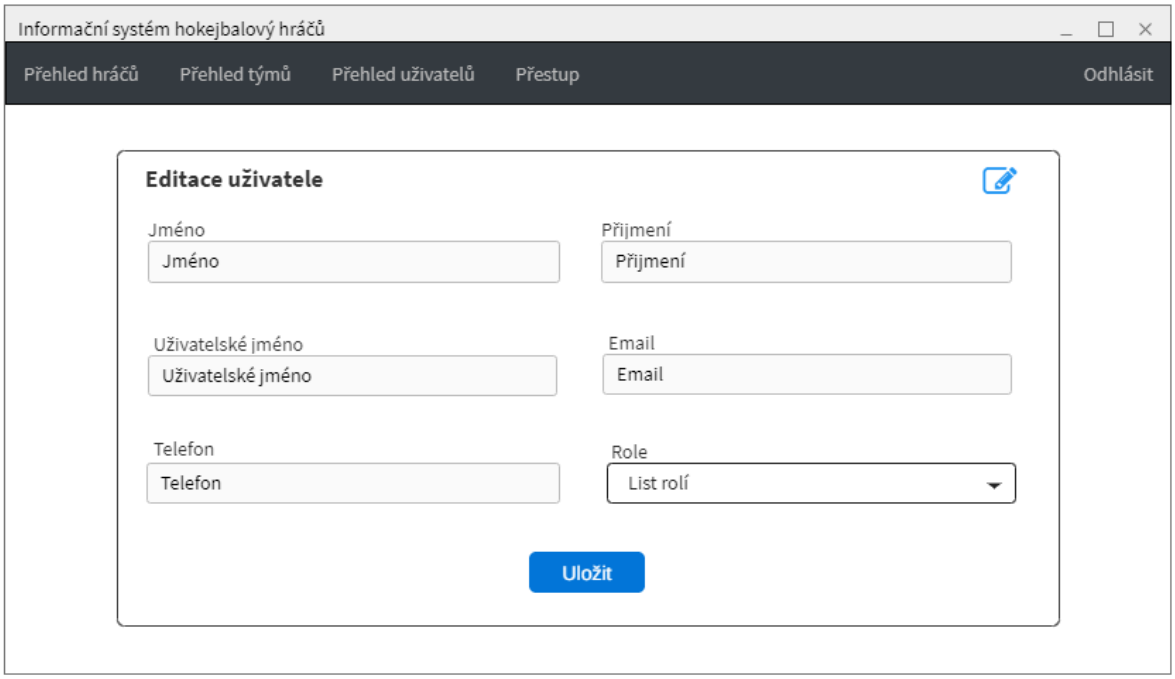
1.9.6 Detail uživatel



Obrázek 12 - Wireframe pro detail uživatele

Stejně jako na stránku pro vyhledání uživatelů mají přístup na stránku pro detail uživatele pouze uživatelé s rolí administrátora. Do detailu uživatele systém vypisuje všechna data kontrétního uživatele. V jednotlivých polích nelze měnit načtená data. V pravé horní části obrazovky se nachází tlačítko pro editaci údajů uživatele.

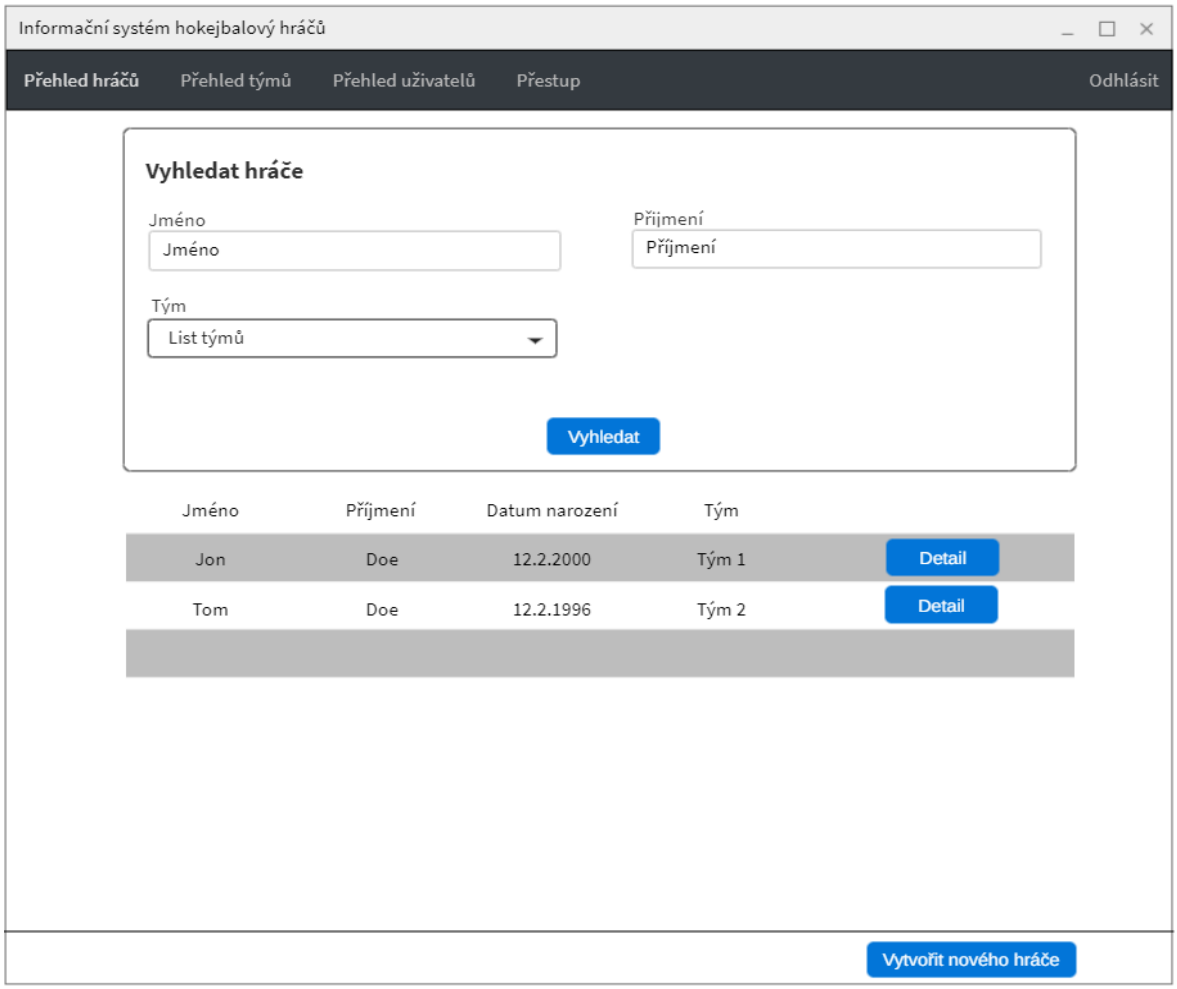
1.9.7 Editace uživatele



Obrázek 13 - Wireframe pro editaci uživatele

V jednotlivých polích jsou vypsána data konkrétního uživatele, která lze přepisovat. V pravé horní části se nachází stejné tlačítko jako na detailu, sloužící k zrušení editace.

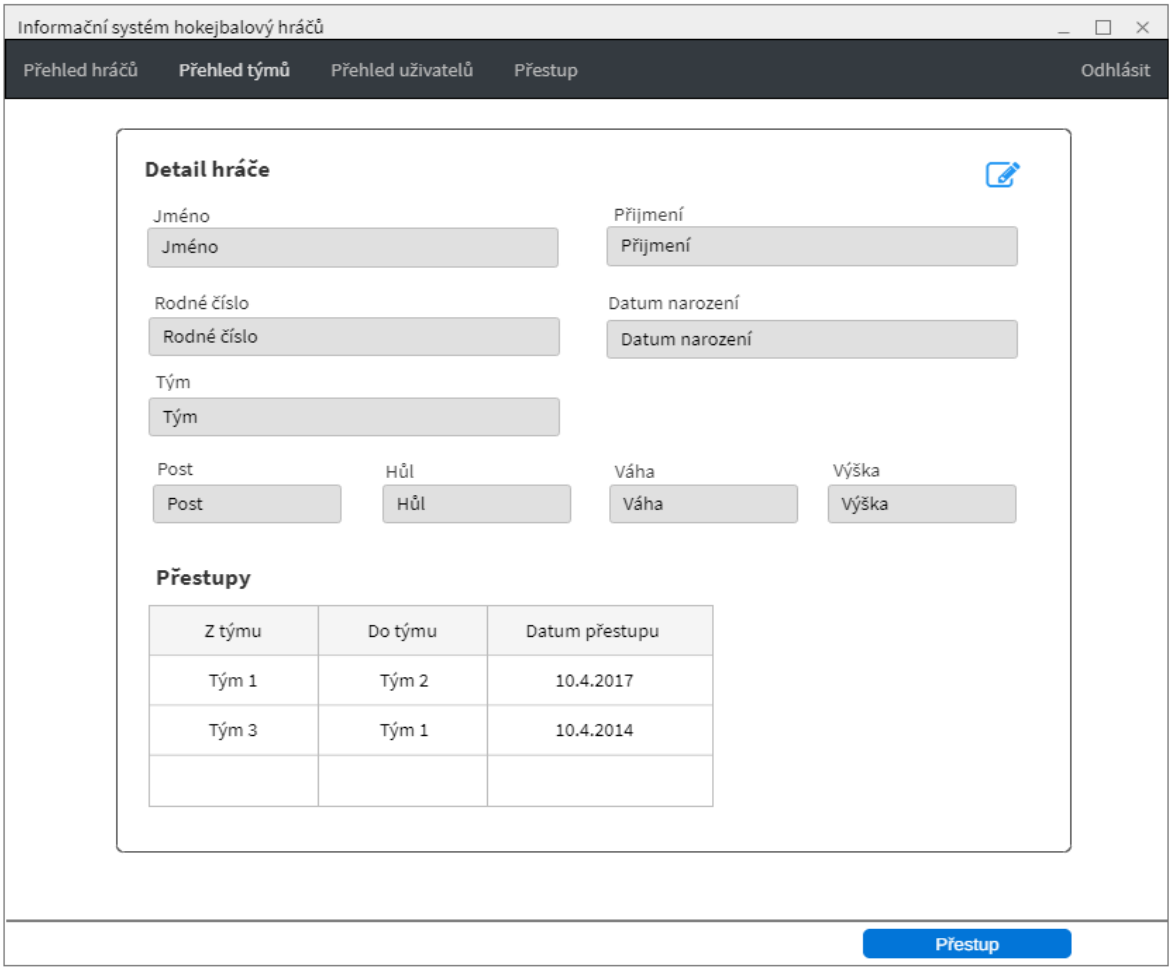
1.9.8 Vyhledání hráče



Obrázek 14 - Wireframe pro vyhledání hráče

Obrazovka pro vyhledání hráčů slouží jako primární obrazovka pro aplikaci a mají na ni přístup všichni uživatelé. V horní části obrazovky se nachází filtr pro vyhledání kontrétního hráče nebo hráčů. V případě vyhledání dat se pod filtr vytvoří tabulka s příslušnými daty a tlačítkem na zobrazení detailu hráče. Pokud se nevyhledají žádná data vypíše se místo tabulky informativní hláška. V dolní části obrazovky se nachází tlačítko „Vytvořit nového hráče“, které je zobrazeno pouze pro uživatele s rolí administrátor nebo editor.

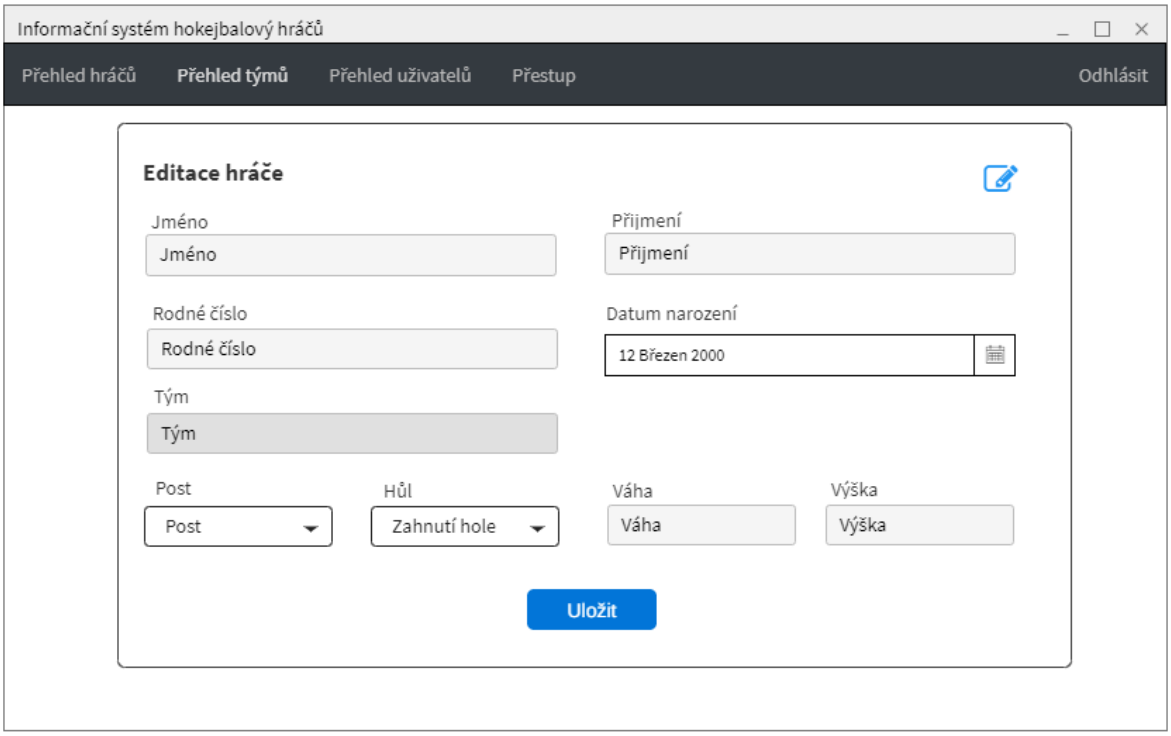
1.9.9 Detail hráče



Obrázek 15 - Wireframe pro detail hráče

Stejně jako na stránku pro vyhledání hráčů mají přístup na stránku pro detail hráče všichni uživatelé. Do jednotlivých polích systém vypisuje všechna data kontrétního hráče. Data v jednotlivých polích nelze měnit a v poli pro rodné číslo je zobrazena pouze informativní hláška, že je rodné číslo nastaveno nebo nenastaveno. V tabulce „Přestupy“ jsou zobrazeny všechny přestupy, které jsou pro konkrétního hráče evidovány. Přes název týmu se lze rovnou prokliknout na jeho detail. Pro uživatele s rolí administrátor nebo editor je horní pravé části zobrazeno tlačítko pro editaci a v dolní pravé části obrazovky tlačítko pro přestup hráče. Výsledný vzhled detailu hráče lze najít v příloze I – Detail hráče.

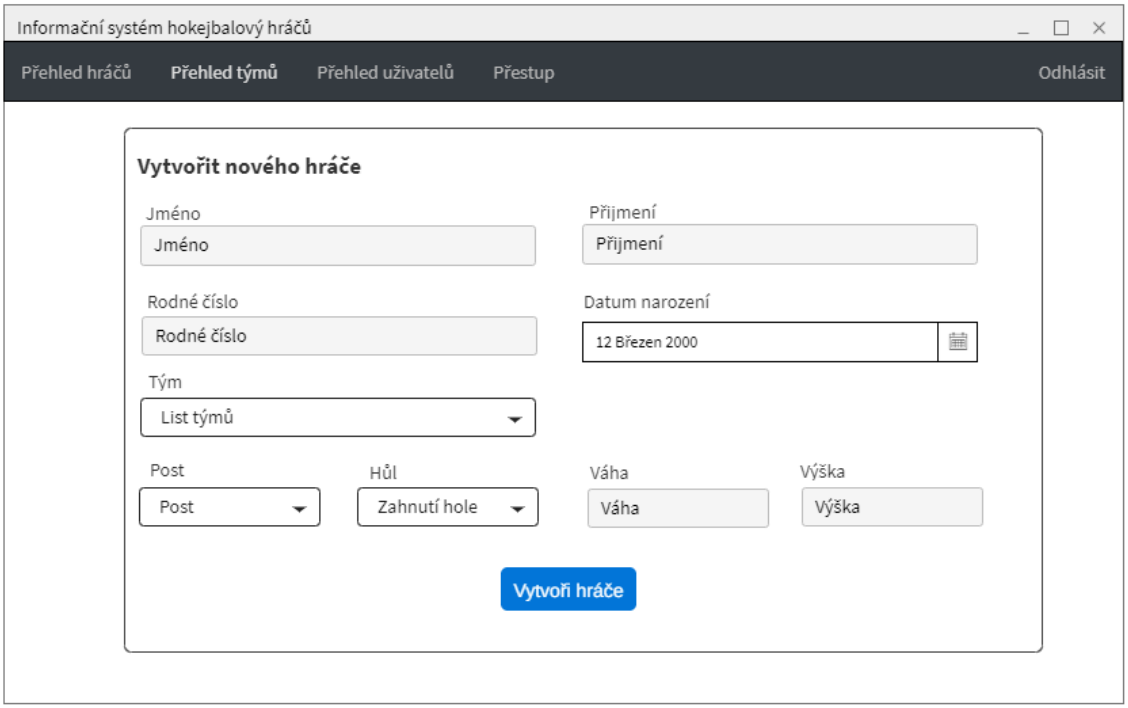
1.9.10 Editace hráče



Obrázek 16 - Wireframe pro editace hráče

Na rozdíl od obrazovky detailu hráče mají na stránku pro editaci přístup pouze uživatelé s rolí editor nebo admin. Při editaci není zobrazena tabulka přestupů a jednotlivá pole se dají měnit, až na pole tým, jelikož změna týmu hráče se provádí pomocí přestupu. V pravé horní části se nachází tlačítko pro zrušení editace hráče.

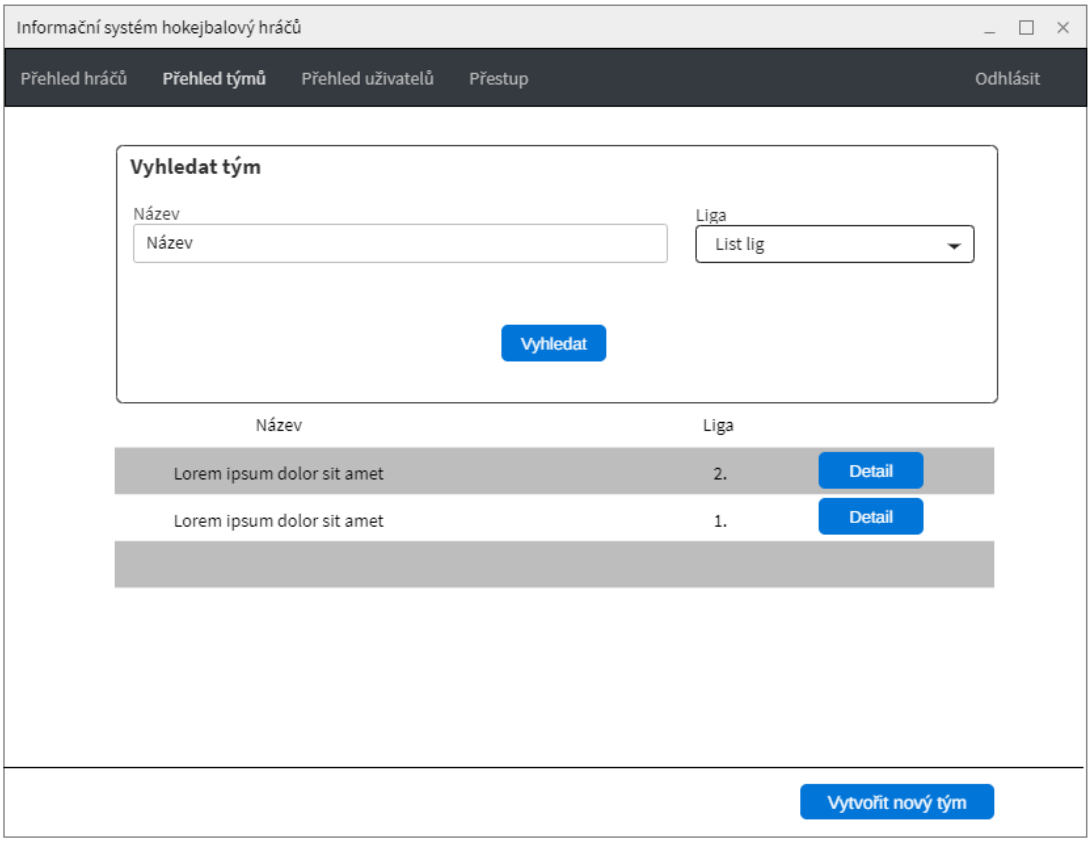
1.9.11 Vytvoření nového hráče



Obrázek 17- Wireframe pro vytvoření hráče

Rozdíl mezi stránkou pro editaci uživatele a vytvoření nového hráče, je možnost výběru týmu, pod který bude hráč zařazen. Až na tuto malou změnu jsou stránky totožné.

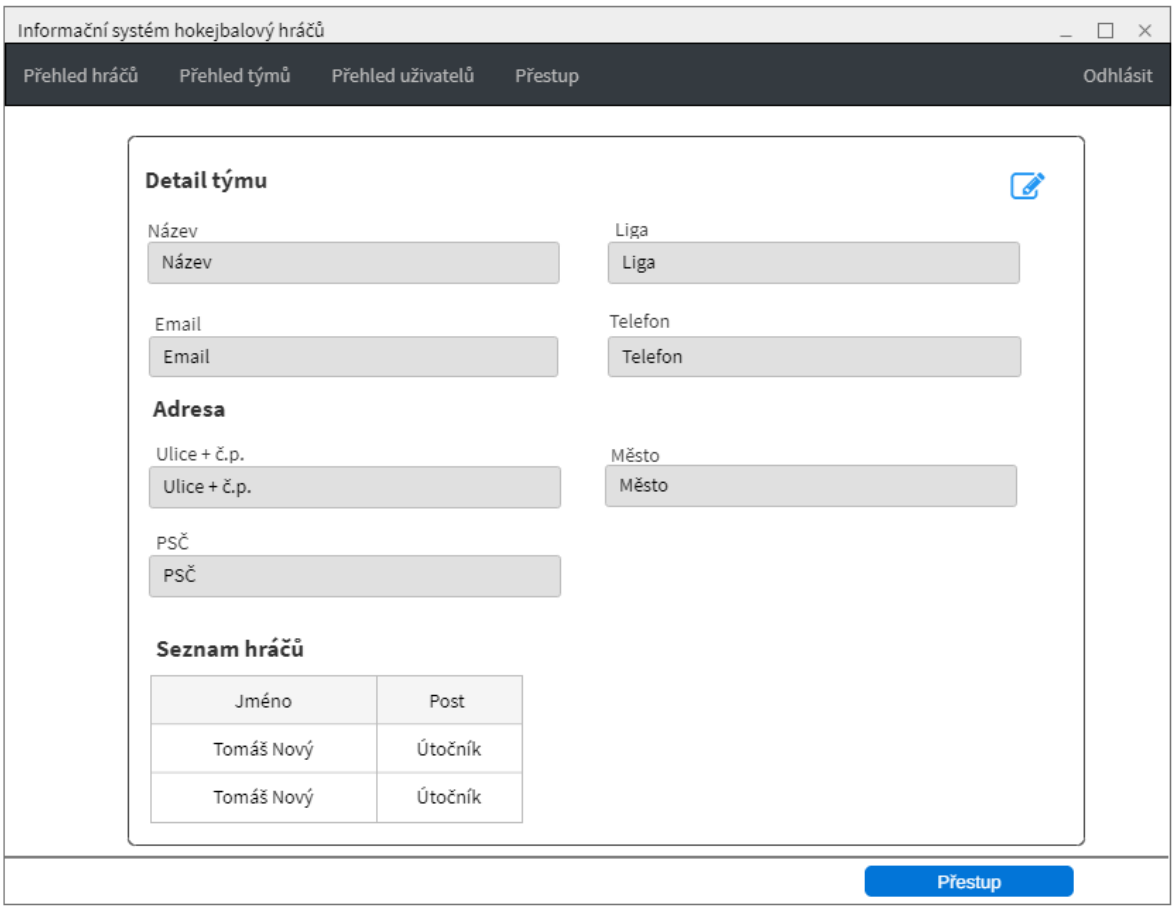
1.9.12 Vyhledání týmu



Obrázek 18 - Wireframe pro vyhledání týmu

Na stránku pro vyhledání týmu mají přístup všichni uživatelé. V horní části obrazovky se nachází filtr pro vyhledání konkrétního týmu nebo týmů. V případě vyhledání dat se pod filtr vytvoří tabulka s příslušnými daty a tlačítkem na zobrazení detailu týmu. Pokud se nevyhledají žádná data vypíše se místo tabulky informativní hláška. V dolní části obrazovky se nachází tlačítko „Vytvořit nový tým“, které je zobrazeno pouze pro uživatele s rolí administrátor nebo editor.

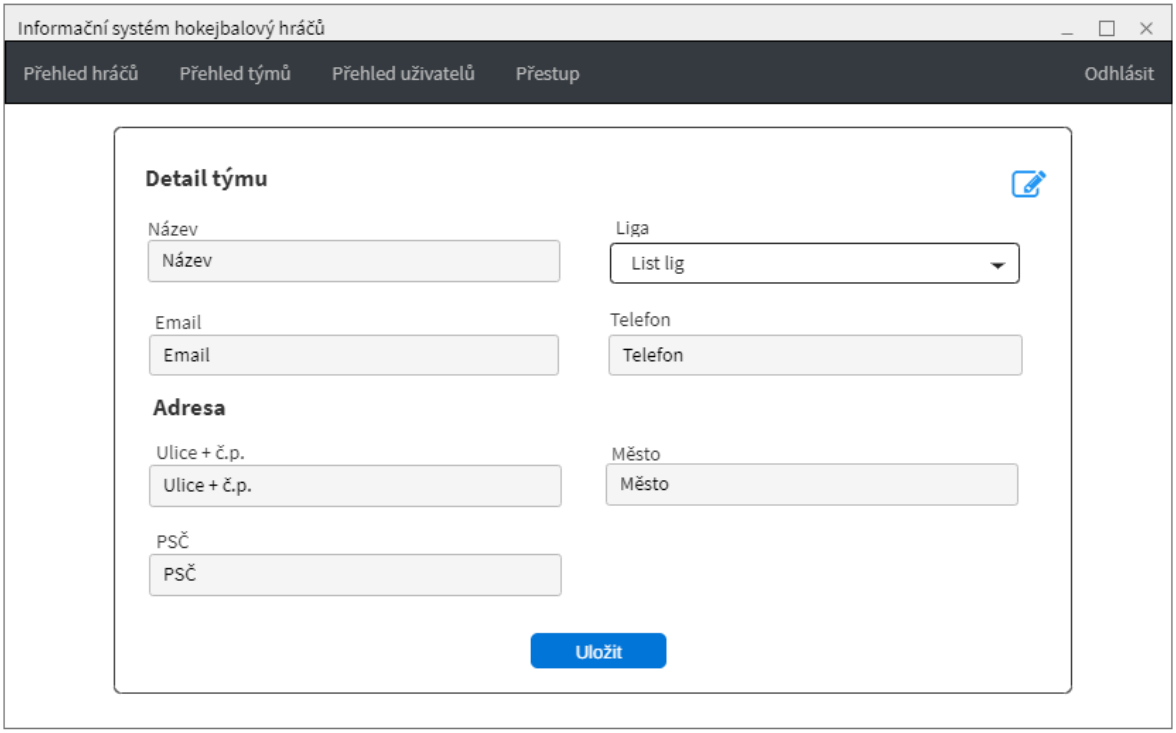
1.9.13 Detail týmu



Obrázek 19 - Wireframe pro detail týmu

Stejně jako na stránku pro vyhledání týmu mají přístup na stránku pro detail týmu všichni uživatelé. Do jednotlivých polích systém vypisuje všechna data konkrétního týmu. Data v jednotlivých polích nelze měnit. V tabulce „Seznam hráčů“ jsou zobrazeni všichni hráči, kteří jsou pro daný tým zaevidováni. Přes jméno hráče se lze rovnou prokliknout na detail konrétního hráče. Pro uživatele s rolí administrátor nebo editor je horní pravé části zobrazeno tlačítko pro editaci a v dolní pravé části obrazovky tlačítko pro přestup do zobrazeného týmu. Výsledný vzhled detailu týmu lze najít v příloze II – Detail týmu.

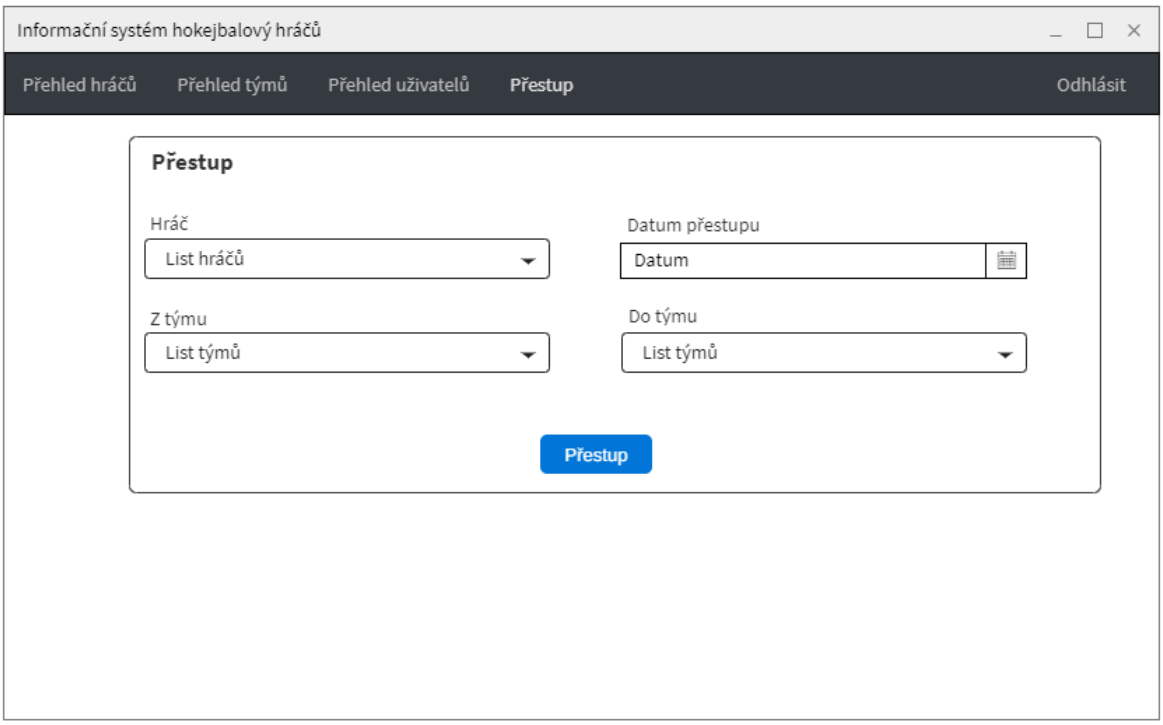
1.9.14 Editace a vytvoření týmu



Obrázek 20 - Wireframe pro editacii týmu

Při vytváření a editaci týmu systém vygeneruje totožný formulář, až na pár změn. Při editaci lze navíc vidět v pravé horní části tlačítko pro zrušení editace, v případě vytvoření týmu se tlačítko na stránce nenachází. Všechna pole na formuláři jsou povinná.

1.9.15 Přestup



Obrázek 21 - Wireframe pro přestup

V případě přístupu na stránku přestupu z menu se výběr v jednotlivých listech dá měnit a nejsou v nich vybrány žádné hodnoty. List hráčů je interaktivní a mění se základě z jakého týmu hráč přestupuje. Jestliže nebude zadáno datum přestupu, tak bude do databáze uloženo datum, kdy byl přestup proveden.

Pokud uživatel přistoupí na tuto stránku z detailu hráče bude vybrán konkrétní hráč a tým ze kterého přestupuje, jediný co lze měnit do jakého týmu hráč přestoupí.

V případě prokliknutí z detailu týmu je vybrána hodnota do jakého týmu hráč přestoupí a lze měnit z jakého týmu a který hráč bude přestupovat.

Výsledný vzhled stránky pro přestup lze najít v příloze III Stránka pro přestup.

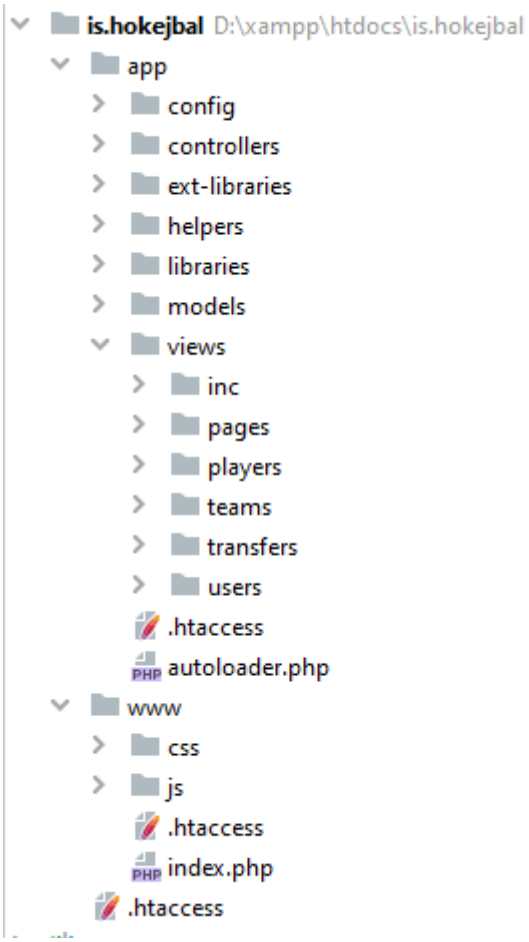
1.10 Implementace

1.10.1 Složková struktura

Na začátku celé implementace byla vytvořena základní složková struktura. Základní strukturu tvoří dvě hlavní složky app a www, které následně obsahují další podsložky a soubory.

První hlavní složka app, obsahuje logiku celé aplikace. Je zde zahrnuta konfigurace aplikace, pomocné třídy například na routování adres a připojení do databáze, externí knihovny a v poslední řadě MVC struktura.

Další hlavní složkou je www, která obsahuje všechny statické soubory jako css a javascript. Dále se zde nachází soubory index.php a .htaccess.



Obrázek 22 - Složková struktura

1.10.2 Routování url adres

1.10.2.1 Nastavení htaccess souborů

První přichází přesměrování celé aplikace do složky www, za pomocí .htaccess souboru v kořenové složce. Díky tomu uživatel místo linku https://domena.cz/www přichází do aplikace za pomocí linku https://domena.cz/.

```
<IfModule mod_rewrite.c>
  RewriteEngine on
  RewriteRule ^$ www/ [L]
  RewriteRule (.*) www/$1 [L]
</IfModule>
```

Zdrojový kód 1 – Nastavení .htaccess souboru pro přesměrování aplikace do složky www

Jako další byl nastaven .htaccess soubor ve složce www, za pomocí kterého je následně vše přesměrováno do souboru index.php, jenž je ve stejné složce. Zde se pak volá třída pro routování adres Router(). Díky nastavení ?url=$1 kde $1 slouží jako placeholder, lze následně zadávat url adresu v přívětivé podobě pro uživatel a to ve tvaru https://domena.cz/players/detail/1.

```
<IfModule mod_rewrite.c>
  Options -Multiviews
  RewriteEngine On
  RewriteBase /is.hokejbal/www
  RewriteCond %{REQUEST_FILENAME} !-d
  RewriteCond %{REQUEST_FILENAME} !-f
  RewriteRule  ^(.+)$ index.php?url=$1 [QSA,L]
</IfModule>
```

Zdrojový kód 2 – Nastavení .htaccess pro přesměrování aplikace do souboru index.php

1.10.2.2 Třída pro routování

Jak je výše zmíněno samotná třída pro routování adres se jmenuje Router(). Třída pro zavolání správného controlleru a metody využívá 4 proměnné.

1. currentController – slouží pro uložení názvu Controlleru, který se má zavolat. Na začátku je nastavena na controller „Players“
2. currentMethod - slouží pro uložení názvu kontrétní metody, která se má v daném Controlleru využít. Na začátku je nastavena na metodu „search“

3. params – list sloužící pro uložení parametrů.
4. homePages – list ve kterém jsou definovány hlavní metody neboli „hlavní stránky“ jednotlivých Controllerů

V první části metody pro routování je provedna kontrola, jestli je vyplněn parametr URL, v případě, že parametr nemá žádnou hodnotu tak se díky počátečnímu nastavení zobrazí stránka pro vyhledání hráčů, která slouží jako hlavní stránka. Pokud parametr vyplněný je, tak se nejdříve zadaná URL metodou explode() rozdělí do podoby listu, kde jako odělovač jednotlivých částí slouží „/“.

Obrázek 23 - Příklad URL adresy od uživatel

```
Array ( [0] => players [1] => detail [2] => 1 )
```

Obrázek 24 - Příklad URL adresy rozdělené do listu po metodě explode()

Po rozdělení URL do listu se currentController nastaví na controller „Pages“ a currentMethod na metodu „error404“, za pomocí tohoto nastavení se při zadaní neexistujícího controlleru nebo metody zobrazí stránka s errorem 404 a informativní hláškou „Je nám líto, ale požadovaná stránka se bohužel na serveru nenachází“.

V dalších krocích se pomocí isset($url[0]) proveda kontrola jestli je nastavena první část URL, pokud je nastavena tak se pomocí metody file_exists() zjistí, jestli zadaný controller z URL existuje. V případě, že zadaný controller existuje nastaví proměnná se currentController na hodnotu zadanou v URL a pomocí unset se z listu smaže hodnota první části URL. V neposlední řadě se kontroluje jestli je po smazání hodnoty list URL prázdný, v případě že je, tak se nastaví currentMethod na metodu z listu homePages na základě controlleru, který je zadán v currentController. Následně se jako currentController nastaví nová instance zadaného controlleru v currentController.

Dále se kontroluje zdali je nastavena druhá část URL pomocí isset($url[1]), v případě, že je, tak se za pomoci metody method_exists(), provede kontrola jestli existuje zadaná metoda v controlleru. Pokud metoda existuje nastaví se proměnná currentMethod na hodnotu zadanou v URL a pomocí unset se z listu smaže hodnota druhé části URL. Pokud zadaná metoda neexistuje nastaví se jako currentMethod nová instance controlleru Pages.

V poslední části kódu se nejdřív do proměnné params nastaví zbylé hodnoty z listu URL. Poté se zavolá třída ReflectionMethod, která obsahuje informace o zadané metodě, konrétně v tomto případě je využita metoda getParameters, která vrací kolik má zadaná metoda vstupních parametrů. V případě, že se nerovná počet parametrů z URL adresy s počtem vstupních parametrů metody, tak se do proměnné currentController nastaví nová instance controlleru Pages a currentMethod se nastaví na metody error404. V neposlední řadě je využita metoda call_user_func_array(), která následně danou metodu zavolá.

Celý kód lze najít v příloze IV Kód pro routování URL adres i s komentáři k jednotlivým částem.

1.10.3 Ochrana před SQL Injection

Pro ochranu před napadením databázové vrstvy aplikace jsou využity takzvané prepared statements neboli připravené příkazy, díky kterým se odesílá šablona a data do databáze zvlášť.

Nejdříve je na server s databází pomocí metody prepare() poslaná šablona dotazu, kde místo jednotlivých dat jsou zástupné symboly. Zástupné symboly mohou být zadány ve formě otazníku nebo lze využít pojmenované parametry, v tomto případě se vloží libovolný název s dvojtečkou na začátku (:název). Databáze dotaz zanalyzuje, provede jeho optimalizaci a uloží výsledek bez provedení.

Poté co je šablona připravena, lze zástupné symboly nahradit daty za pomocí metody bind(). Po nastavení dat následuje spuštění dotazu pomocí metody execute().

```php
$this->dbh = new PDO($dsn, $this->user, $this->pass, $options);

$name = 'Test';

$sql = 'SELECT * FROM test WHERE name = :name';
$stmt = $this->dbh->prepare($sql);
$stmt->bindValue(':name', $name);
$stmt->execute();
```

Zdrojový kód 3 – Ukázka exekuce příkazů za pomocí prepared statements

1.10.4 Zabezpečení hesel

Pro bezpečné ukládání hesel do databáze je využita metoda password_hash(), do které vstupují dva parametry. Prvním parametr je heslo v podobě textu, které uživatel zadal. Druhým parametrem je název hashovací algoritmu, který má být využit. V PHP existuje konstanta PASSWORD_DEFAULT, která lze zadat jako název hashovacího algoritmu, tato konstanta je automaticky aktualizována vždy na nejbezpečnější algoritmus. Z důvodu automatické aktualizace jsem se zmíněnou konstantu rozhodl využít ve své aplikaci. Metoda následně vrací hash zadaného hesla, který je následně uložen do databáze.

```php
$data['passwordHash'] = password_hash($password, PASSWORD_DEFAULT);
```

Zdrojový kód 4 - Vytvoření hashe hesla

Pro následnou validaci hesla je využita metoda password_verify(), do které vstupují dva parametry. Prvním parametrem je zadané heslo od uživatele v podobě textu, druhý parametr je uložený hash v databázi. Funkce následně porovná jestli se hash právě zadaného hesla shoduje s hashem uloženým v databázi.

```php
password_verify($password, $hashed_password)
```

Zdrojový kód 5 – Validace hesla

1.10.5 Knihovna na odesílání emailů

Samotné PHP obsahuje funkci na odesílání emailů mail(), ovšem tato funkce je velmi omezená. Hlavní nevýhodou této funkce je nemožnost nastavit odesílaní přes SMTP server, následně odeslaný email nemá protokol TLS. .Další nevýhodou je nemožnost přípojit k emailu přílohu.

Z výše uvedených důvodů je využita knihovna PHPMailer (github), která je distribuovaná pod licencí GNU/LGPL v2.1. Za pomocí této knihovny lze například

jednoduše k emailům přidávat přílohu nebo zasílat email na několik adres najednou. Největší výhodou je ovšem možnost odesílání emailů přes zadaný SMTP server.

1.10.6 Návrh databáze

Za účelem uchovávání většího množství dat, byla vytvořena databázová strukura, která obsahuje 9 entit. Z celkového počtu 9 entit jsou 4 číselníkové. Samotný model databáze lze najít v příloze V Návrh databáze.

1.11 Testování aplikace

Po implementaci celého systému přišlo na řadu testování. Na začátku byly sestaveny základní funkčnosti, které jsou pro chod aplikace ty nejnutnější. Mezi ty funkčnosti patří.

1. Přihlášení
2. Vyhledání hráču
3. Zobrazení detailu hráče
4. Vytvoření hráče
5. Vyhledání týmu
6. Zobrazení detailu týmu
7. Vytvoření týmu
8. Vyhledání uživatele
9. Zobrazení detailu uživatele
10. Vytvoření uživatele

Následně na tyto funkčnosti byly napsány automatické testy pro pravidelné testování. Zbylé funkčnosti byly otestovány manuálně jako přestup, změna hesla, nastavení trvalého hesla a jednotlivé editace byly otestovány manuálně.

1.11.1 Automatické testy

Jak je výše zmíněno, pro automatické testování bylo vytvořeno celkem 10 testů. Automatické testy byly v průběhu implementace puštěny několikrát. Celková doba trvání testů je přibližně 50 sekund a výsledky lze najít v příloze VI Výsledek automatických testů

V případě zakládání nové entity do systému, test kontroluje jestli se v databázi vytvořil nový záznam. Následně se validuje zdali data uložená v databázi souhlasí s daty, která byla zadaná při vytváření. Po zkontrolování správného uložení záznamu v databázi je záznam smazán.

Při kontrolování jednotlivých detailů se vyhledá libovolný, již existující záznam v databázi. Podle vyhledaného id test přejde na kontrétní detail záznamu. V tomto případě se nejdříve kontroluje, zdali jsou pole na obrazovce disabled a následně se provede kontrola dat zadaných na obrazovce oproti datům zadaných v databázi.

1.12 Nasazení aplikace

Po úspěšném otestování aplikace, přišlo na řadu nasazení aplikace na server. Od 13.03.2021 je aplikace k dispozici na stránce http://client.mdns.cz/is.hokejbal/. Pro otestování všech funkčností lze využít účet vytvořený pro automatické testování aplikace.

Přihlašovací údaje:

- Uživatelské jméno: AD22
- Heslo: admin

5. Závěr

Hlavním tématem bakalářské práce bylo vytvoření informačního systému pro správu hokejbalových hráčů a jejich týmů.

V teoretické části byly popsány metodiky, které lze pro vývoj aplikace využít, po nichž následoval popis životního cyklu vytváření softwaru.

Praktická část je věnována popisu celého vývoje aplikace. Všechny stanovené funkčnosti, které byly sestaveny pomocí use case požadavků byly splněny.V rámci práce taktéž byly vytvořeny automaticky testy, za jejichž pomocí se dají pravidelně testovat nejzákladnější funkce aplikace. V neposlední řadě došlo k nasazení aplikace na server, díky čemuž je informační systém dostupný ze všech zařízení.

Přesto, že byly splněny všechny funkčnosti, které byly na začátku sestaveny je možností jak samotný systém rozšířit nepřeberné množství. Jako první se nabízí funkčnost na zadávání zápasů mezi jednotlivými týmy, které jsou ve stejné lize. Následně by se na základě jednotlivých výsledků mohla aktualizovat výsledná tabulka soutěže.

Dále by se dalo vytvořit API pomocí, kterého by mohly jednotlivé týmy dotahovat data o hráčích na své webové prezentace. Týmy by si mohly například jednou týdně pomocí definovaného API rozhraní dotáhnout aktualizovaná statistiky hráčů a nemusely by je zdlouhavě zadávat ručně, obzvlášť pokud tým disponuje více věkovými kategoriemi týmů.

V poslední řadě by se hodilo logování jednotlivých akcí do databáze, díky tomu by se dalo jednoduše dohledat jaký uživatel, kterému měnil role nebo kdo upravoval již uložené entity za pomocí aplikace.

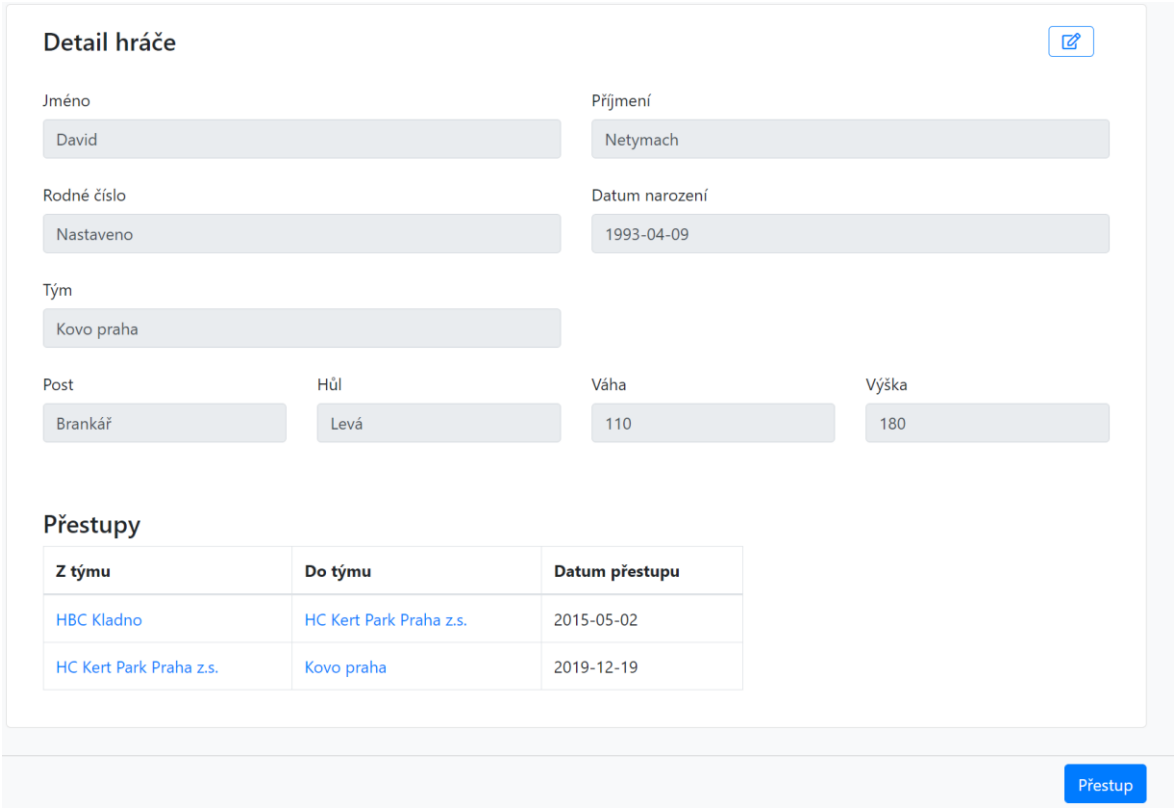
6. Seznam použitých zdrojů

1. Vývojový cyklus software: Dominují agilní metodiky trhu?. Middleware.cz - IT v souvislostech [online]. Copyright © 2015 [cit.07.03.2021]. Dostupné z: https://www.middleware.cz/projektove-rizeni0/24-pristupy-k-vyvoji-software-dominuje-agilni-vyvoj-trhu
2. SDLC - Waterfall Model - [online]. Copyright © Copyright 2021. All Rights Reserved. [cit.09.03.2021]. Dostupné z: https://www.tutorialspoint.c om/sdlc/sdl c_waterfall_ m odel.htm
3. **THE AGILE ALLIANCE.** Manifest agilního vývoje software: Agile Manifesto [online]. Copyright © 2001 [cit. 10.03.2021] Dostupné z: http://www.agilemanifesto.org/iso/cs/manifesto.html
4. **Martin Večeřa**. 5 Metod Vedení Projektů: 3. Díl – Scrum a Kanban. Lumeer [online]. Copyright © Lumeer.io s.r.o. Založeno 2017. [cit. 10.03.2021]. Dostupné z: https://www.lumeer.io/cs/scrum-a-kanban-metoda/
5. **Claire Drumond.** Scrum - what it is, how it works, and why it's awesome[online]. Copyright © 2021 Atlassian [cit. 10.03.2021]. Dostupné z: https://www.atlassian.com/agile/scrum
6. What is SDLC? Phases of Software Development & Models. phoenixNAP: Data Center, Dedicated Servers, Cloud, & Colocation [online]. Copyright © 2021 Copyright phoenixNAP [cit. 14.03.2021]. Dostupné z: https://phoenixnap.com/blog/software-development-life-cycle
7. Vývoj softwaru - atlantis telecom | atlantis software. Home - atlantis telecom | atlantis software [online]. Copyright © 1994 [cit. 09.03.2021]. Dostupné z: https://www.atlantis.cz/vyvoj-softwaru
8. MVC Framework - Introduction [online]. Copyright © Copyright 2021. All Rights Reserved. [cit. 14.03.2021]. Dostupné z: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
9. **Martin Malý**. Rest: architektura pro webové API [online]. [cit. 10.03.2021]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

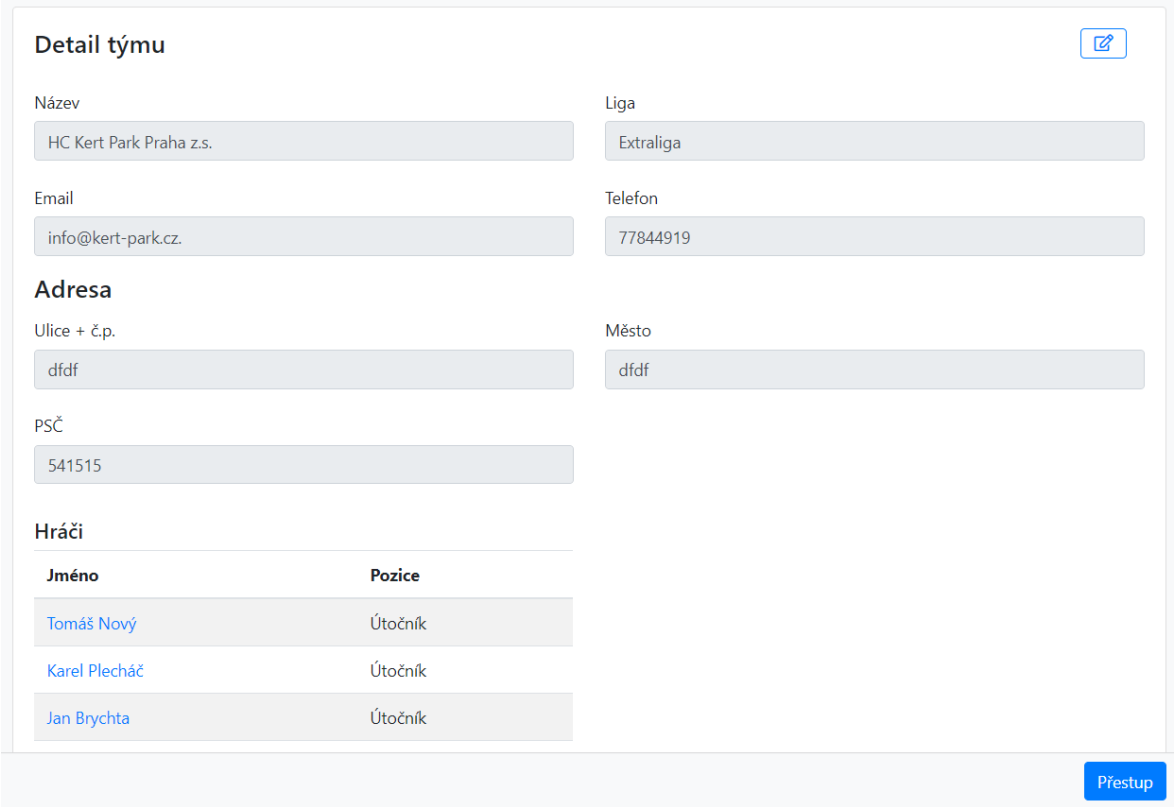
10. **Anna Monus.** SOAP vs REST vs JSON – a 2020 comparison [online]. Copyright © Copyright Raygun 2018 [cit. 10.03.2021]. Dostupné z: https://raygun.com/blog/soap-vs-rest-vs-json/#differences
11. **Wendell Santos.** Which API Types and Architectural Styles are Most Used? [online]. ProgrammableWeb, 2017. [cit. 10.03.2021] Dostupné z : https://www.programmableweb.com/news/which-api-types-and-architectural-styles-are-most-used/research/2017/11/26
12. **Kingthorin.** SQL Injection | OWASP. OWASP Foundation, the Open Source Foundation for Application Security [online]. [cit. 10.03.2021] Dostupné z: https://owasp.org/www-community/attacks/SQL_Injection
13. What is Bootstrap [online]. [cit. 10.03.2021]. Dostupné z: https://whatis.Techtarget.com/definition/bootstrap
14. **WELLING, Luke a Laura THOMSON**. Mistrovství PHP a MySQL. Přeložil Ondřej BAŠE. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.
15. Robot Framework. [online]. [cit. 10.03.2021]. Dostupné z : https://robotframework.org/
16. What is Python? Opensource.com [online]. Copyright ©2021 [cit. 10.03.2021]. Dostupné z: https://opensource.com/resources/python

7. Přílohy

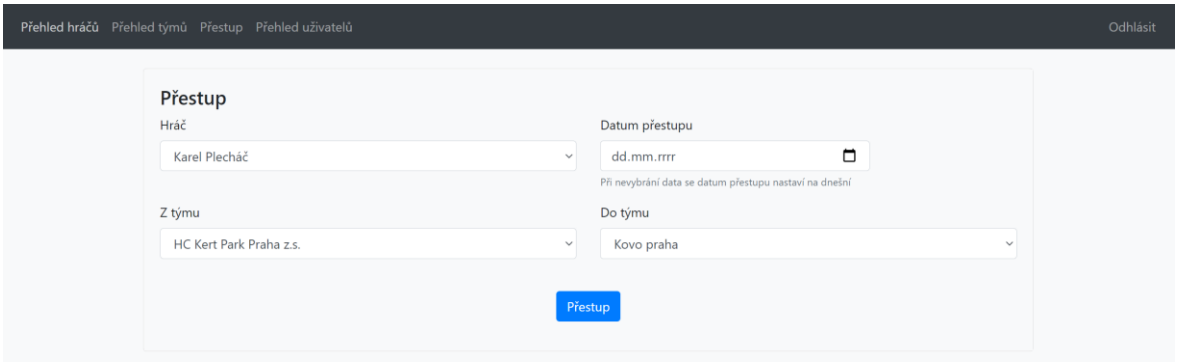
Příloha I: Detail hráče



Příloha II: Detail týmu



Příloha III: Stránka pro přestup

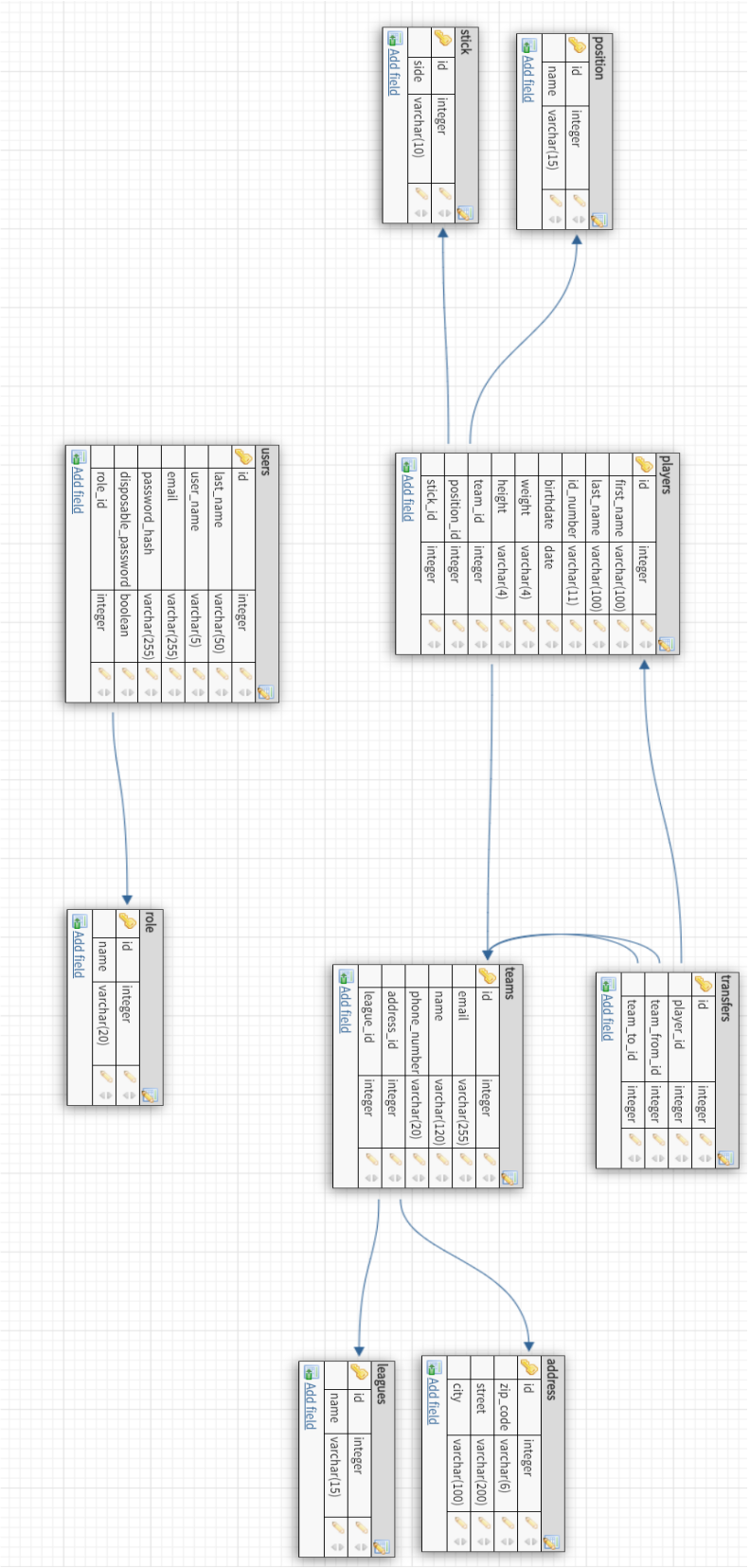


Příloha IV: Kód pro routovaní URL adres

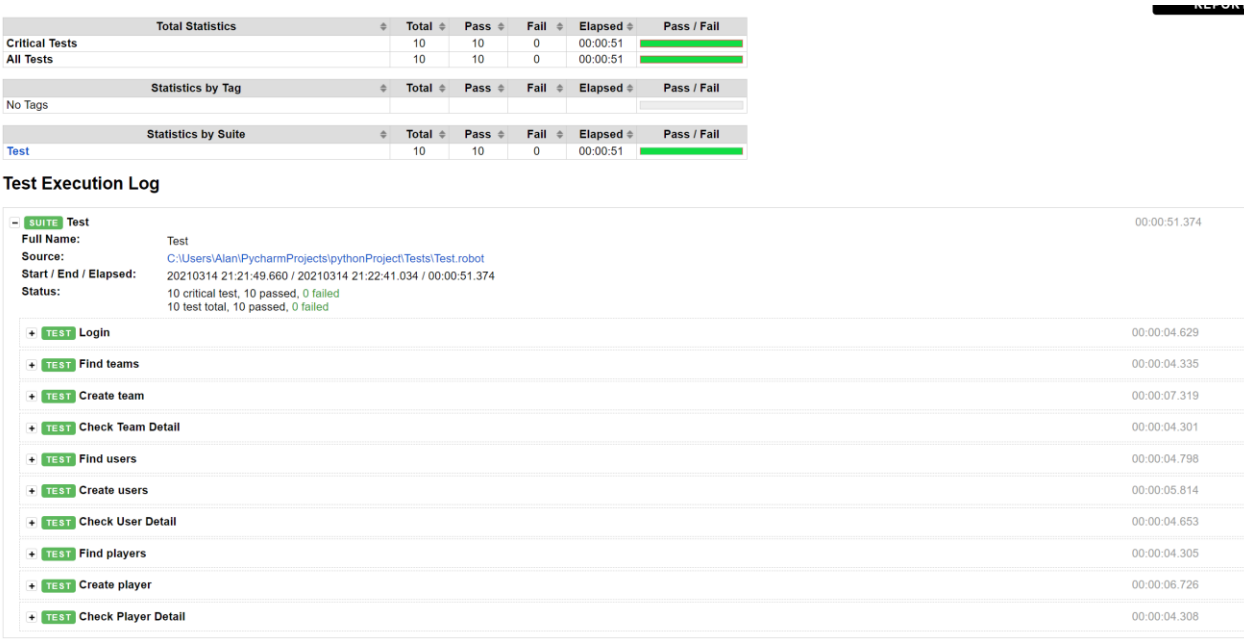
```
class Router extends Controller
{
    //nastavení hlavního controlleru a metody
    protected $currentController = 'Players';
    protected $currentMethod = 'search';
    protected $params = [];

    // nastavení hlavních metod
    protected $homePages = [
        'Players' => 'search',
        'Transfers' => 'transfer',
        'Teams' => 'search',
        'Users' => 'search'
    ];

    public function __construct()
    {
        $url = '';
        if (isset($_GET['url'])) {
            //rozdělí zadanou url do podoby listu
            $url = explode('/', $_GET['url']);
            //nastavení currentController a currentMethod na Error stránku
            $this->currentController = 'Pages';
            $this->currentMethod = 'error404';
        }

        //kontrola jestli je nastavena první část URL
        if (isset($url[0])) {
            //kontrola existence controlleru
            if (file_exists('../app/controllers/' . ucwords($url[0]) . '.php')) {
                //nastavení currentControlleru na hodnotu zadanou v URL
                $this->currentController = ucwords($url[0]);
                // smaže z listu první část URL
                unset($url[0]);
            }
            //kontroluje jestli je URL prázdná
            if(empty($url)) {
                //nastavení currentMethod na hodnotu z listu homePages podle
proměnné currentController
                $this->currentMethod= $this->homePages[$this->currentController];
            }
        }

        //za proměnou currentController se dosadí nová instance controlleru
        $this->currentController = new $this->currentController;

        //kontrola jestli je nastavena druhá část URL
        if (isset($url[1])) {
            //kontrola jestli existuje v controlleru
            if (method_exists($this->currentController, $url[1])) {
                //pokud metoda existuje dosadí se do proměnné currentMethod
hodnota z druhé části url
                $this->currentMethod = $url[1];

                //smaže z listu druhou část URL
                unset($url[1]);
            } else {
                //pokud metoda neexistuje za proměnou currentController se dosadí
nová instance controlleru Pages
                $this->currentController = new Pages();
            }
        }
```

```
        // zbylé hodnoty z URL nastaví jako list parametrů
        $this->params = $url ? array_values($url) : [];
        //zavolá se instance třídy ReflectionMethod, díky které se zjistí
informace o metodě
        $method = new ReflectionMethod($this->currentController, $this-
>currentMethod);


        //kontrola jestli počet parametrů zadaných v URL souhlasí s počtem
vstupních parametrů metody
        if (count($this->params) != count($method->getParameters())) {
            //pokud ne tak se nastaví currentController na novou instantci třídy
Pages
            $this->currentController = new Pages();
            //currentMethod se nastaví na metodu error404
            $this->currentMethod = 'error404';
        }

        //zavolá metodu se zadanými parametry
        call_user_func_array([$this->currentController, $this->currentMethod],
$this->params);
    }
}
```

Příloha V: Návrh databáze



Příloha VI: Výsledek automatických testů



Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	10	10	0	00:00:51	
All Tests	10	10	0	00:00:51	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Test	10	10	0	00:00:51	

Test Execution Log

SUITE Test 00:00:51.374

Full Name: Test
Source: C:\Users\Alan\PycharmProjects\pythonProject\Tests\Test.robot
Start / End / Elapsed: 20210314 21:21:49.660 / 20210314 21:22:41.034 / 00:00:51.374
Status: 10 critical test, 10 passed, 0 failed
10 test total, 10 passed, 0 failed

- TEST Login 00:00:04.629
- TEST Find teams 00:00:04.335
- TEST Create team 00:00:07.319
- TEST Check Team Detail 00:00:04.301
- TEST Find users 00:00:04.798
- TEST Create users 00:00:05.814
- TEST Check User Detail 00:00:04.653
- TEST Find players 00:00:04.305
- TEST Create player 00:00:06.726
- TEST Check Player Detail 00:00:04.308